ELSEVIER

Computer Networks and ISDN Systems 29 (1997) 675-699

COMPUTER NETWORKS and ISDN SYSTEMS

An analytic study of partially ordered transport services [1]

Rahmi Marasli [2], Paul D. Amer*, Phillip T. Conrad [3]

Computer and Information Sciences Department, University of Delaware, Newark, DE 19716, USA

Accepted 30 October 1996

Abstract

This paper presents an analytic model for investigating the throughput, delay and buffer utilization characteristics of partially ordered transport services. We analyze the effects of packet and ack losses as well as applications' order requirements on overall system performance. The analytic model is verified by comparing its results against those of an OPNET simulation model. Analytic results show that for applications that can tolerate some reordering in the delivery of objects, use of partially ordered service instead of ordered service provides important buffer utilization and delay improvements, particularly as the loss rate increases and the order requirements of applications decrease. In terms of throughput, it makes no difference which service (i.e., ordered, partially ordered, unordered) an application uses. Analytic study also shows that by judicious choice of sender's transmission order, overall system performance can further be improved in a partially ordered service. © 1997 Elsevier Science B.V.

Keywords: Transport layer protocol; Protocol design and analysis; Partially ordered service; Quality of service; Multimedia

1. Introduction

Computer networks traditionally offer either ordered (e.g., TCP) or unordered (e.g., UDP) transport service. Some applications such as multimedia do not need an ordered service since they can tolerate some reordering in the delivery of the objects. The degree of reordering should be within the specific limits of the applications; otherwise problems result at the application layer such as increased complexity, increased buffering, and loss of synchronization. For such applications, neither ordered nor unordered service is a perfect fit. Ordered service insists on delivering all data in sequence even if it results in higher delays and buffer utilization. Unordered service, on the other hand, minimizes delay and buffer utilization, but provides no order guarantees. If an application with some order constraints uses an unordered transport service, the application programmer is burdened with the task of implementing mechanisms for object ordering.

* Corresponding author. Email: amer@cis.udel.edu.

[1] This work supported, in part, by the National Science Foundation (NCR-9314056), the US Army Communication Electronics Command (CECOM), Ft. Monmouth, the US Army Research Office (DAAH04-94-G-0093, DAAL03-92-G-0070), and the US Department of the Army, Army Research Laboratory under Cooperative Agreement DAAL01-96-2-0002 Federated Laboratory ATIRP Consortium.

[2] Email: rmarasli@fore.com.

[3] Email: pconrad@cis.udel.edu.

PII S0169-7552(96)00113-4

To achieve better tradeoffs between order and other quality-of-service (QoS) parameters, and to satisfy the minimal order requirements of applications, partially ordered transport service has been proposed [1,3,6]. Partially ordered service fills the gap between ordered and unordered service by allowing applications to specify the delivery order of objects in the form of a partial order. Since partially ordered service does not insist on delivering all objects in sequence, it can provide lower delays and buffer utilization than ordered service, while, at the same time, guaranteeing an application's partial order requirements.

The authors are designing a new transport-layer protocol, called Partial Order Connection (POC), that provides *partially ordered* and *partially reliable*[4] service to its users [1,3,6]. POC enhances an unreliable/unordered network service *just enough* to allow applications to specify *controlled* levels of loss and reordering in the delivery of the objects. Thus, both the order and the reliability requirements of the applications are generalized in POC. Previous study has formally confirmed the intuitive results that, in general, a *partially reliable* service provides lower delay and higher throughput than a reliable service [10]. This paper analytically studies the *partially ordered* aspect of POC. This analytic study basically has the following goals:

(1) To obtain quantitative measures on how well partially ordered transport service performs over various network environments, and to gain understanding of how various network and application parameters (e.g., loss level of the network layer, sender's transmission order, order requirements of the applications, etc.), in general, affect system performance.
(2) To show the performance improvements by using partially ordered service over ordered service, and thus, motivate the use of partially ordered service against ordered service.

The paper is organized as follows: Section 2 introduces a partially ordered service and motivates it with two example applications. Section 3 introduces an analytic model for partially ordered service and discusses computational results. The analytic computations are verified by an OPNET-based simulation in Section 4, and the main results are summarized in Section 5.

2. Why use a partially ordered service?

Refs. [1,6] introduce the development and motivation for a partially ordered protocol/service including several examples. A summary of these findings is provided here.

Essentially, a partially ordered 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. In general, the order requirements of objects in a partially ordered service can be represented by using a partial order PO over the set $[N] = \{1, 2, \ldots, N\}$, where N is the total number of objects to be communicated, and $x \prec y$ in PO signifies that object x must be delivered to the receiving application prior to object y.

2.1. A simple application for partially ordered service: Screen refresh

Consider an application that does a "screen refresh" on a workstation screen/display containing multiple windows (see Fig. 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. Objects that do not overlap may be refreshed in any order. Therefore, the way in which the windows overlap induces a partial order.

[4] Partial reliability refers to the notion that individual objects may have different QoS requirements with respect to loss; some may require reliable transport service (guaranteed no-loss), while for others, unreliable transport service (best-effort) may suffice. Partially reliable transport service provides a middle ground between these two in which the loss tolerance of each object can be specified individually. References [1,3,4,6] consider partial order and partial reliability in juxtaposition, while this paper focus solely on partial order.

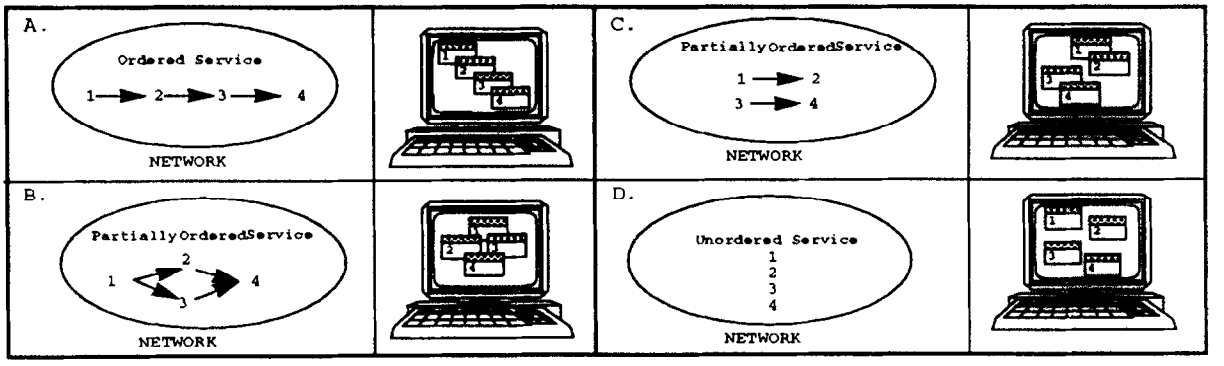


Fig. 1. Screen refresh.

Consider the four cases in Fig. 1. A sender wishes to refresh a remote display that contains four active windows (objects) named {1234}. Assume that the windows are transmitted in numerical order and that the receiving application refreshes windows as soon as the transport layer delivers them. If the windows are configured as seen in Fig. 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 due to loss or disorder in the network layer, window 2 is received before window 1, the transport layer must buffer window 2 and deliver it only after window 1 arrives and is delivered.

At the other extreme, if the windows are configured as in Fig. 1(D), an unordered service would suffice. Here any of 4! delivery orderings would satisfy the application since the four windows can be refreshed in any order. Each of these orderings represents a linear extension (*LE*) of the partial order (*PO*). 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). Figs. 1(B) and 1(C) demonstrate two (of many) window configurations that call for a partial order delivery service. In these cases, two and six linear extensions, respectively, are permitted at the destination.

2.2. *Using partially ordered service for remote document retrieval*

Ref. [4] describes a prototype client/server application for the retrieval and display of multimedia documents from a remote server using Partial Order Connection version 2 (POCv2), a partially ordered and partially reliable transport protocol providing coarse-grained synchronization support. In this system, multimedia documents with temporal characteristics are described using a Prototype Multimedia Specification Language (PMSL). PMSL gives an author the ability to specify the synchronization, (partial) order, and (partial) reliability requirements of the objects that make up a temporal multimedia document. The application serving these documents can extract these requirements from such a specification and communicate them to the transport layer, which then provides the necessary QoS and synchronization support.

This simplifies application development, since the document display client need not contain complex mechanisms for object synchronization and reordering. It also allows for graceful degradation, since the document can be presented "perfectly" when network conditions allow, and in a less than perfect but nevertheless acceptable manner when network conditions degrade. Finally, the use of partial order and partial reliability rather than ordered/reliable or unordered/unreliable service allows better QoS tradeoffs between qualitative parameters such as order/reliability and quantitative parameters such as delay, buffer utilization and throughput.

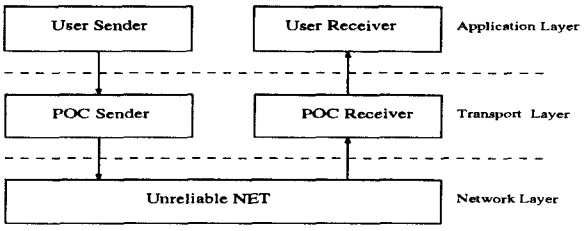


Fig. 2. Architecture.

3. Analytic model

In this section, we present an analytic model for partially ordered transport services. Through this model, we study the effects of packet and ack losses as well as various levels of applications' order requirements on the performance of different services (i.e., ordered, partially ordered, and unordered). Verification of the analytic model using simulation will be described later in Section 4.

The analytic study confirms our expectations that a partially ordered service provides lower delay than an ordered service while simultaneously requiring less buffer space at the receiver. Results also show that in a partially ordered service, the choice of the sender's transmission order further impacts overall system performance.

3.1. Introduction to model

To abstract partially ordered service's usage, we use a three layer architecture which includes only the network layer, the transport layer, and the user application layer (see Fig. 2).

The transport layer protocol provides a partially ordered service as follows: POC Sender takes a packet from User Sender, transmits the packet over the network, then sets a timer and buffers the packet. If the corresponding ack does not return from POC Receiver within its timeout period, POC Sender retransmits the packet. When a packet is received at POC Receiver, if the packet is deliverable (i.e., if all packets that this packet depends on have already been delivered), then it is delivered to User Receiver; otherwise it is buffered. Upon delivering any packet, POC Receiver checks its buffers for additional packets that may have become deliverable as a result of the delivery; these packets are delivered also.

By assumption, User Sender submits constant size packets to POC Sender. In general, given a partial order with variable object sizes, we can obtain an equivalent partial order with constant object size by fragmenting large objects into smaller constant size ones that are chained to each other.[5] Thus, having constant packet sizes in the computations should not limit the effectiveness of our results.

In the network layer (called Unreliable NET), the loss of a packet or an ack is characterized by a Bernoulli process, and a constant end-to-end network delay is assumed. It is also assumed that POC Receiver never runs out of buffer space. The full set of assumptions about the system in general can be found in Table 2. These simplifying assumptions, while in some cases are strong (e.g., Bernoulli losses), are needed for the mathematical analysis of the model.[6] The results obtained under these assumptions are useful in *comparing* various types of services, and in analyzing *trends*. We expect the effects of these assumptions to be similar across various levels of services (i.e., ordered, partially ordered, unordered). These expectations are, in part, supported by simulation results (see Sections 3.4.1 and 3.5).

[5] The last fragment may have to be padded.

[6] Even with these simplifying assumptions, the mathematical analysis is complicated and we have only approximate computations for certain target values.

Table 1
System variables

System variables	Definitions
t_{pack}	packet transmission time
t_{delay}	one-way delay for a packet defined as "t_{pack} + one-way network layer delay"
RT	round trip delay defined as "t_{pack} + (2 * one-way network layer delay + ack transmission time)"
t_{out}	timeout period for retransmissions
p	probability of losing a packet within Unreliable NET
q	probability of losing an ack within Unreliable NET
Buf_S	number of buffers at POC Sender
Buf_R	number of buffers at POC Receiver
p_{succ}	probability of successful packet and ack transmission defined as "$(1-p)*(1-q)$"

Table 2
Assumptions

	Assumptions
1	p and q are fixed and independent for each packet and ack transmission
2	RT is constant and $t_{out} = RT$
3	Packets and acks have constant sizes
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	$Buf_S = t_{out}/t_{pack}$ and $Buf_R = \infty$
7	Only selective acks are used
8	All packets are ready at User Sender, or equivalently, a packet arrives at User Sender at every t_{pack}
9	User Sender submits packets to POC Sender in an order that respects the given PO

The system variables are defined in Table 1. Throughout this paper, we refer to this system as *NET*. Thus,

$$NET = \langle t_{pack}, t_{delay}, RT, t_{out}, p, q, Buf_S, Buf_R, p_{succ}, A \rangle,$$

where $t_{pack}, \ldots, p_{succ}$ represent the system variables, and A stands for the assumptions given in Table 2. Unless otherwise stated, all subsequent values and computations in this paper refer to this given *NET* with linear extension *LE* of partial order *PO* being used as POC Sender's transmission order.

3.2. Definitions

In this paper, we analyze the throughput, delay and buffer utilization characteristics of partially ordered transport services. This analysis is done by computing the performance statistics defined in Table 3. Throughput, λ, is the rate at which POC Receiver delivers packets to User Receiver. End-to-end packet delay, T_{end_a}, is the expected time for packet a to reach to User Receiver once it is given to POC Sender. For many applications such as real time audio and video, lower delay is more important than higher throughput. Finally, expected buffers used at the receiver, $\overline{R\_Buff}$, indicates the average memory resources utilized at POC Receiver. In general, it is desirable to have higher λ, lower T_{end_a}, and lower $\overline{R\_Buff}$.

In addition to the performance statistics of Table 3, we also compute the buffering probabilities and buffering times for a partially ordered service. For this, Table 4 defines four target values. The investigation of buffering probabilities and times is done for three reasons. First of all, we need to know $Buf_{a,b}$ and Buf_a in order to compute T_{end_a} and $\overline{R\_Buff}$. Secondly, the analysis of buffering characteristics helps us better understand the overall analytic model. For example, we introduce some approximations to the computations of T_{end_a} and $\overline{R\_Buff}$. These approximations are easier to understand when explained through buffering probabilities. Finally, for packets a and b such that $a \prec b$ in PO, we investigate the negative effects of packet a's loss on packet b through the target values $pBuf_{a,b}$ and $Buf_{a,b}$.

Table 3
Important performance statistics

Throughput (λ)	Average number of packets delivered to User Receiver per unit time
End-to-end Packet Delay (T_{end_a})	Expected end-to-end delay for packet a
Buffers Used at Receiver ($\overline{R\_Buff}$)	Average number of packets buffered at POC Receiver waiting to be delivered to User Receiver

Table 4
Buffering probabilities and times

$pBuf_{a,b}$	P(packet a arrives after packet b) if $a \prec b$; 0 otherwise
$Buf_{a,b}$	E(time that packet b is buffered waiting for a to arrive at POC Receiver) if $a \prec b$; (Note: if there is no constraint $a \prec b$, then $Buf_{a,b} = 0$)
$pBuf_a$	P(packet a is buffered at POC Receiver)
Buf_a	E(time that packet a is buffered at POC Receiver)

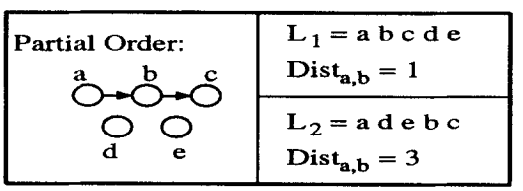


Fig. 3. $Dist_{a,b}$ values for two different linear extensions of a partial order.

Let $Dist_{a,b}(L)$ be the distance between packets a and b in the linear extension L defined as "seq(b) − seq(a)" where seq(x) returns the assigned sequence number for packet x in L. Notice that $Dist_{a,b}$ values can be different for different linear extensions of a PO. As an example, consider the PO in Fig. 3 and two of its linear extensions L_1 and L_2. The distance between packets a and b for these two linear extensions are 1 and 3, respectively. In our computations, we use the $Dist_{a,b}$ values of the dependent pairs (i.e., $a \prec b$ in PO) to express the LE and PO information in the target values.

In our analysis, we first present the computations of buffering probabilities and times in Section 3.3. We then introduce the investigation of Table 3's performance statistics (i.e., throughput, end-to-end packet delay and expected buffers used at receiver) in Section 3.4. We study the effects of using a different LE as the sender's transmission order on system performance in Section 3.5.

3.3. Analysis of buffering probabilities and times

This section presents the investigations of buffering probabilities and times for a partially ordered service. This analysis proceeds by first computing these target values for a dependent pair of packets (i.e., $pBuf_{a,b}$ and $Buf_{a,b}$). We then expand our computations to general buffering probabilities and times (i.e., $pBuf_a$ and Buf_a) for any given packet.

3.3.1. Buffering probability and time between dependent pairs: $pBuf_{a,b}$ and $Buf_{a,b}$

In a partially ordered service, there are packets whose delivery depends on other packets having been delivered. For example, for packets a and b such that $a \prec b$ in PO, packet b cannot be delivered to User Receiver unless packet a has already been delivered. This is the requirement needed to assure that transport protocol provides the application's desired partially ordered service. Hence, if b is received before a, then b should be buffered at POC Receiver until after a's arrival and delivery. In this section, we study the buffering effects on b of a's loss when $a \prec b$ in PO.

$pBuf_{a,b}$ is defined as the probability of having to buffer b due to loss of a. Let $pSB_{a,b}$ be the probability that all transmissions of packet a (i.e., original transmission and any retransmissions) preceding the first transmission

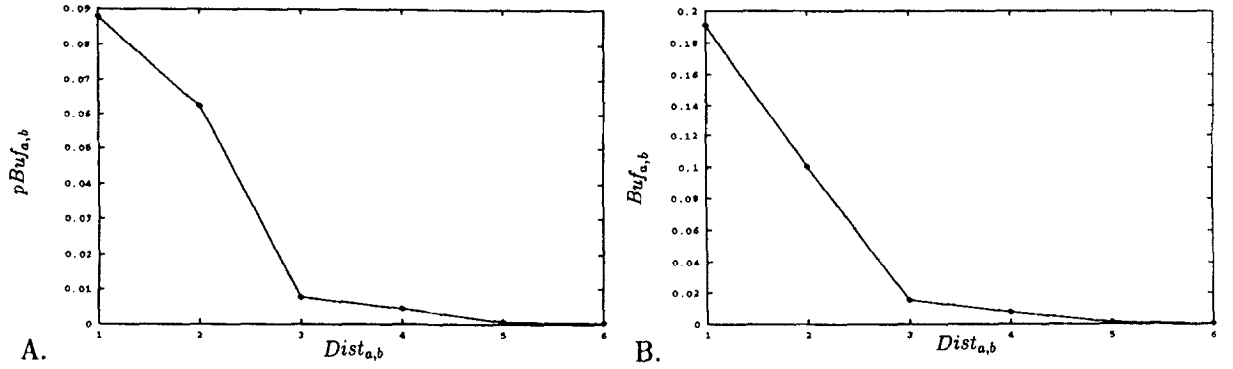


Fig. 4. Effects of $Dist_{a,b}$ on (A) $pBuf_{a,b}$ and (B) $Buf_{a,b}$ when $p = q = 0.1$, $Buf_S = 3$ and $t_{pack} = 1$.

of packet b fail. Then we can compute $pBuf_{a,b}$ by using $pSB_{a,b}$ as follows:[7]

$$pBuf_{a,b} = \frac{1}{1+p} * pSB_{a,b} \tag{1}$$

Similarly, $Buf_{a,b}$, the expected time that packet b is buffered waiting for packet a to arrive, can be computed as follows:

$$\begin{aligned} Buf_{a,b} &= pSB_{a,b} * \frac{(1-p) * (Buf_S - SL_{a,b}) * t_{pack} + p * t_{out}}{1-p^2} \\ &= pBuf_{a,b} * \frac{(1-p) * (Buf_S - SL_{a,b}) * t_{pack} + p * t_{out}}{1-p} \end{aligned} \tag{2}$$

where $SL_{a,b}$ is the expected number of slots between a's failure just before b's first transmission, and b's first transmission (see Appendix A.3 for the computational details of $Buf_{a,b}$ and $SL_{a,b}$).

Figs. 4(A) and 4(B) illustrate "$pBuf_{a,b}$ and $Buf_{a,b}$ vs $Dist_{a,b}$" for the system configuration where $p = q = 0.1$, $Buf_S = 3$, and $t_{pack} = 1$. Note that $t_{pack} = 1$ represents the normalized case where it takes one unit time to transmit a packet. These graphs show that both target values decrease with increasing $Dist_{a,b}$ values. Thus, for any two packets a and b where $a \prec b$ in PO, the negative effects of a's loss on b decrease as the separation of b from a in the transmission order of the packets (i.e., $Dist_{a,b}$) increases. It is noteworthy that at large $Dist_{a,b}$ values (e.g., $Dist_{a,b} \geqslant 5$), the buffering of b due to the loss of a is almost totally eliminated.

This is an encouraging result since by putting some distance between the transmission orders of dependent packets, POC Sender can significantly reduce buffering probabilities and times in the system. That is, by wisely deciding on the packet sending order, POC Sender can improve the overall system performance. After computing all target values, we will further discuss the importance of sending transmission order in Section 3.5.

3.3.2. *Buffering probability for a packet: $pBuf_a$*

$pBuf_a$ is the overall buffering probability for packet a. Note that as the buffering probabilities in a system decrease, buffering times also decrease. Since packets are buffered for shorter times, end-to-end packet delays and buffer utilization at the receiver should tend to be smaller. Thus, overall system performance should improve as $pBuf_a$ decreases. In this section, we investigate the conditions for this target value to decrease.

Notice that packet a will be buffered if and only if it overtakes (i.e., arrives before) a packet b and $b \prec a$ in PO. Then:

[7] Computational details for $pSB_{a,b}$ and $pBuf_{a,b}$ can be found in Appendix A.1 and A.2, respectively.

$$pBuf_a = P\left(\bigcup_{\forall b \prec a} a \text{ overtakes } b\right)$$

$$= \sum_{\forall b \prec a} P(a \text{ overtakes } b) - \sum_{\forall \{b1,b2\} \prec a} P\left(\bigcap_{k=1}^{2} a \text{ overtakes } bk\right)$$

$$+ \sum_{\forall \{b1,b2,b3\} \prec a} P\left(\bigcap_{k=1}^{3} a \text{ overtakes } bk\right) - \cdots \quad (3)$$

$$P\left(\bigcap_{k=1}^{m} a \text{ overtakes } bk\right) = pSB_{b1,\ldots,bm,a} * \left(\frac{1-p}{1-p^{m+1}}\right) \quad (4)$$

where $pSB_{b1,\ldots,bm,a}$ is the probability that all transmissions (original plus retransmissions) of packets $b1$ through bm preceding the first transmission of packet a fail.[8] Appendix A presents an exact computation for $pSB_{b,a}$. We only have an approximate expression for $pSB_{b1,\ldots,bm,a}$ when $m \geqslant 2$ in [8]. Thus, besides being complicated, the buffering probability for packet a does not have an exact expression.

In general, we can simplify (3) by using a different approximation. The terms $P(\bigcap_{k=1}^{m} a \text{ overtakes } bk)$ decrease with decreasing Buf_S and p. Therefore, we do not have to compute all of the terms in expression (3) under low loss rates and sender buffer sizes. Under such situations, $pBuf_a$ can be approximated as:

$$pBuf_a \simeq \sum_{\forall b \prec a} P(a \text{ overtakes } b) = \sum_{\forall b \prec a} pBuf_{b,a} \quad (5)$$

Intuitively, we can explain this approximation as follows: when the loss rate is low, it is unlikely that packet a would overtake two or more preceding packets. Hence, when p is small, the buffering probability for packet a can be approximated by expression (5) since $pBuf_a \simeq \sum_{\forall b \prec a} P(a \text{ overtakes } b) = \sum_{\forall b \prec a} pBuf_{b,a}$. Similarly, if Buf_S is small, then fewer number of packets b such that $b \prec a$ have a chance of failing until the first transmission of packet a. Thus, packet a has a smaller chance of overtaking two or more preceding packets. Therefore, expression (5) will be a good approximation to buffering probabilities when Buf_S or p is small.[9] It is noteworthy that when $Buf_S = 2$, $pBuf_a$ will be exactly equal to expression (5) since all terms $P(\bigcap_{k=1}^{m} a \text{ overtakes } bk) = 0$ for $m \geqslant Buf_S$.

Expression (5) shows that as the number of packets b such that $b \prec a$ decreases, or $Dist_{b,a}$ values increase,[10] $pBuf_a$ will decrease. In general, if the density of PO is small[11] (i.e., relatively few ordering constraints), there will be a smaller number of packets b such that $b \prec a$. Additionally, $Dist_{b,a}$ values between dependent packets can in general be made larger by POC Sender's choice of transmission order in the case of low density POs. Combining these two observations, we can say that if PO has low density, then the buffering probabilities in the system will be lower.

Fig. 5 introduces three partial orders with densities 0.4, 0.6 and 1.0, respectively. Throughout this paper, we use these three partial orders in periodic form while comparing the performance of different services (i.e., partially ordered service with different densities, ordered service) with each other. A *periodic PO* is defined as a partial order repeating itself some number of times. Periodic POs can be represented as $P \oplus \cdots \oplus P$ where

[8] See [8] for approximate computations of $pSB_{b1,\ldots,bm,a}$ and $P(\bigcap_{k=1}^{m} a \text{ arrives before } bk)$.

[9] We will approximate the computations of Buf_a, T_{end_a} and $\overline{R\_Buff}$ under the same conditions.

[10] $pBuf_{b,a}$ decreases with increasing $Dist_{b,a}$; see Fig. 4(A).

[11] The *density* of a partial order is a measurement defined as follows [7]. 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.

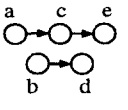
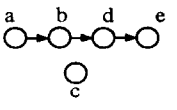
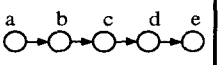
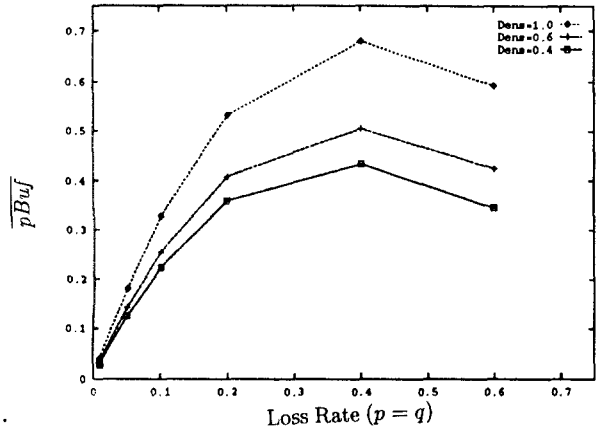
1. Partial order with Density=0.4	2. Partial order with Density=0.6	3. Partial order with Density=1.0
		

Fig. 5. Partial orders with different densities.



A.

B.

	Base Densities			Improvements over Chain	
$p = q$	0.4	0.6	1.0	0.4	0.6
0.01	0.03	0.03	0.04	30.3%	20.2%
0.05	0.13	0.14	0.18	31.1%	21.0%
0.10	0.22	0.26	0.33	31.7%	22.0%
0.20	0.36	0.41	0.53	32.7%	23.5%
0.40	0.43	0.51	0.68	36.3%	25.7%
0.60	0.34	0.42	0.59	41.8%	28.5%

Fig. 6. Average buffering probabilities (i.e., $\overline{pBuf}$) with three different *POs* when $Buf_S = 5$ and $t_{pack} = 1$.

P is the base partial order repeatedly concatenated to itself.[12] The *base density* of a periodic *PO* is defined as the density of just one period. Notice that for a chain *PO*, the base density (or the density of any number of periods) will be equal to 1. Thus, the third *PO* in Fig. 5 represents an ordered service, while the other two represent partially ordered services with periodic *POs* having base densities 0.4 and 0.6. For these three *POs*, $LE = abcde$ is used as transmission order.

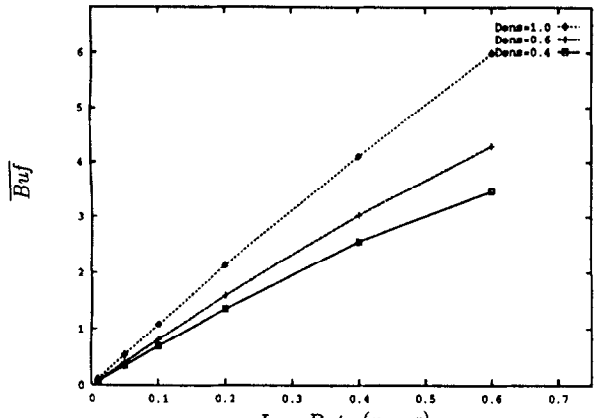
Let $\overline{pBuf}$ be the average of buffering probabilities defined as $\overline{pBuf} = (\sum_{i=1}^{N} pBuf_i)/N$, where N is the total number of packets. Fig. 6(A) illustrates $\overline{pBuf}$ values for the three partial orders shown in Fig. 5. Similarly, the table in Fig. 6(B) introduces the corresponding values for $\overline{pBuf}$. The rightmost two columns of the table give the percentage improvements in buffering probabilities by using either the 0.4 or the 0.6 base density *PO* instead of the chain (i.e., the 1.0 density). As an example, at 0.1 loss level (i.e., $p = q = 0.1$), $Buf_S = 5$ and $t_{pack} = 1$, the average of buffering probabilities for the 0.4 base density *PO* is 0.22 and this is 31.7% smaller than that of the chain.

Fig. 6 clearly shows that as the density of *PO* decreases, buffering probabilities in the system decrease. This figure also shows that the improvements in $\overline{pBuf}$ increase with increasing loss rate. For example, while the absolute[13] and the percentage gains by the 0.4 base density *PO* are 0.16 and 32.7%, respectively, at 0.2 loss level, they increase to 0.25 and 36.3% at 0.4 loss level. It is noteworthy that the absolute gains by partially ordered services are negligible at small loss rates (e.g., $p, q < 0.05$). This is because buffering probabilities by ordered service are already low and there is not much to improve by using partially ordered services.

In general, we can conclude that a partially ordered service provides important buffering probability gains over an ordered service when the density of *PO* is low and the loss rate is high.

[12] "$\oplus$" is the linear sum or concatenation operator for *POs* defined [5] as $x \prec y$ in $P \oplus Q$ if and only if $x, y \in P$ and $x \prec y$ in P, or $x, y \in Q$ and $x \prec y$ in Q, or $x \in P$ and $y \in Q$.

[13] Absolute $\overline{pBuf}$ gain simply refers to the difference between the $\overline{pBuf}$ values of a partially ordered service and those of an ordered service.



A.

B.

	Base Densities			Improvements over Chain	
$p = q$	0.4	0.6	1.0	0.4	0.6
0.01	0.06	0.07	0.10	39.0%	27.3%
0.05	0.34	0.40	0.53	36.9%	25.7%
0.10	0.69	0.81	1.08	36.2%	25.0%
0.20	1.36	1.60	2.13	36.3%	24.9%
0.40	2.55	3.03	4.09	37.7%	26.0%
0.60	3.47	4.28	5.99	42.1%	28.5%

Fig. 7. Average buffering times (i.e., $\overline{Buf}$) with three different *POs* when $Buf_S = 5$ and $t_{pack} = 1$.

3.3.3. *Expected buffering time for a packet: Buf_a*

This section investigates the buffering times characteristics of partially ordered services. Buf_a is the expected time that packet a spends at the buffers of POC Receiver. In general, lower buffering times achieve desirable lower delay and buffer utilization.

We already discussed during the computation of $pBuf_a$ that when Buf_S or p is small, it is unlikely that two or more packets will be overtaken by a later packet. Using this fact, we approximate Buf_a as follows:

$$Buf_a \simeq \sum_{\forall b \prec a} E(\text{time that } a \text{ waits for } b) = \sum_{\forall b \prec a} Buf_{b,a}. \tag{6}$$

Since with low density *POs*, the number of packets b such that $b \prec a$ will be smaller, and $Dist_{b,a}$ values between dependent pairs can be made larger,[14] based on expression (6), we can say that buffering times decrease with decreasing densities of *POs*. This can easily be seen in Fig. 7 that illustrates the average of buffering times (i.e., $\overline{Buf} = (\sum_{i=1}^{N} Buf_i)/N$) for the three *POs* in Fig. 5. Fig. 7 also shows that as the loss rate increases, while the percentage gains change only slightly, the absolute gains increase.

Based on the results of Section 3.3, we can conclude that for applications that do not need an ordered service, by using partially ordered service, buffering probabilities and times in the system can significantly be reduced especially at high loss rates.

3.4. *Analysis of performance statistics*

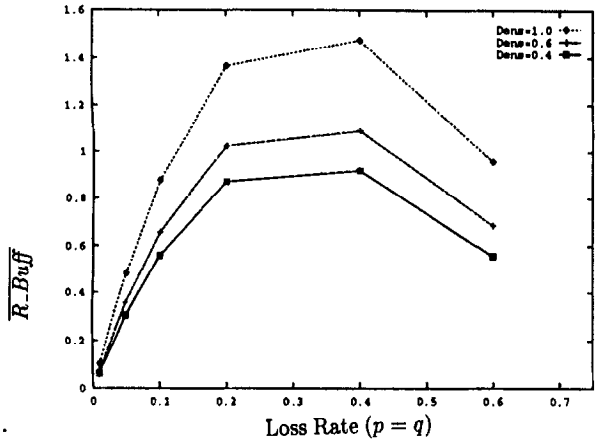
The main objective of this analytic study is to investigate the throughput, delay and buffer utilization characteristics of partially ordered transport services. In general, our analysis proceeds as follows:

(1) We compute λ, throughput, using Little's theorem.
(2) The formula for $\overline{R\_Buff}$, expected buffers used at receiver, is derived by using buffering times and Little's theorem.
(3) T_{end_a}, end-to-end delay for packet a, is computed by using the expression for buffering times.

3.4.1. *Throughput: λ*

With the assumptions 2, 3, 4, 5, 6, and 8 of Table 2, the number of packets at POC Sender is always Buf_S. Let PS_{Time} be the expected time that a packet spends at the buffers of POC Sender. By Little's theorem:

[14] $Buf_{b,a}$ decreases with increasing $Dist_{b,a}$; see Fig. 4(B).



A.

B.

	Base Densities			Improvements over Chain	
$p = q$	0.4	0.6	1.0	0.4	0.6
0.01	0.06	0.07	0.10	39.0%	27.3%
0.05	0.30	0.36	0.48	36.9%	25.7%
0.10	0.56	0.66	0.88	36.2%	25.0%
0.20	0.87	1.02	1.36	36.3%	24.9%
0.40	0.92	1.09	1.47	37.7%	26.0%
0.60	0.56	0.69	0.96	42.1%	28.5%

Fig. 8. Expected buffers used at receiver (i.e., $\overline{R\_Buff}$) with three different POs when $Buf_S = 5$ and $t_{pack} = 1$.

$Buf_S = \lambda * PS_{Time}$. PS_{Time} can be computed as "$PS_{Time} = \sum_{i=1}^{\infty} i * t_{out} * p_{succ} * (1 - p_{succ})^{i-1} = t_{out}/p_{succ}$". Thus, since $Buf_S = t_{out}/t_{pack}$ by assumption 6, throughput is:

$$\lambda = \frac{p_{succ}}{t_{pack}} \text{ packets/unit time.} \tag{7}$$

Expression (7) shows the interesting result that throughput does not depend on the order requirements of applications (i.e., PO). Notice that we obtain this result under the assumptions of infinite buffers at receiver (i.e., $Buf_R = \infty$) and constant network layer delays. Thus, based on expression (7), we can conclude that a partially ordered service does not provide a throughput improvement over an ordered service when $Buf_R = \infty$ and network layer delay is constant.

Would this analytic result be valid if we relax these two assumptions? The simulation results from [8] show that partially ordered services provide a throughput improvement only when POC Sender has more buffers than POC Receiver (i.e., $Buf_S > Buf_R$). Thus, even though our analytic result is derived under constant network layer delays and infinite buffers at receiver, it also holds for variable network layer delays and for finite Buf_R such that $Buf_R \geqslant Buf_S$.

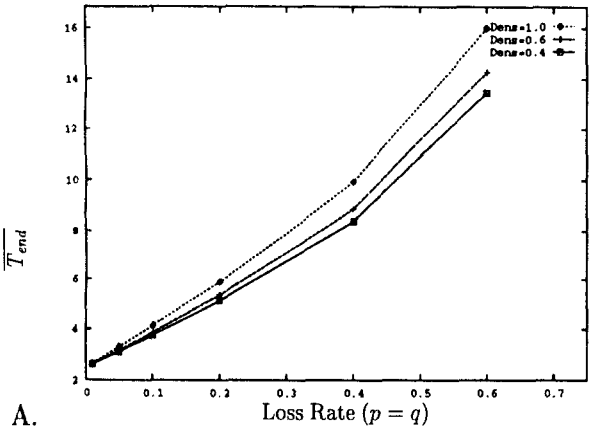
Since most transport layer protocols tend to use sender and receiver buffer sizes of roughly equal size, for most practical purposes, we can conclude that a partially ordered service provides no throughput improvement over an ordered service. Thus, our analytic result explains the relationship between throughput and order requirements of applications for most practical cases.

3.4.2. *Buffers used at Receiver:* $\overline{R\_Buff}$

In general, a packet spends $\overline{Buf} = (\sum_{i=1}^{N} Buf_i)/N$ time on average at POC Receiver. Using Little's theorem, $\overline{R\_Buff}$, the expected number of packets buffered at POC Receiver waiting to be delivered to User Receiver, can be computed as:

$$\overline{R\_Buff} = \lambda * \overline{Buf} \text{ packets} \tag{8}$$

Fig. 8 illustrates $\overline{R\_Buff}$ values for the three POs in Fig. 5. It shows that buffer utilization is lower with lower densities of POs. As an example, at 0.1 loss level, $\overline{R\_Buff}$ with 0.4 and 0.6 base density POs are respectively 36.3% and 24.9% smaller than that of ordered service (i.e., 1.0 density). Fig. 8 also shows that while there are significant percentage improvements in $\overline{R\_Buff}$ at all loss levels, the absolute gains are negligible at small loss rates (e.g., $p, q < 0.1$).



A.

B.

	Base Densities			Improvements over Chain	
$p = q$	0.4	0.6	1.0	0.4	0.6
0.01	2.61	2.63	2.65	1.5%	1.1%
0.05	3.10	3.16	3.30	6.0%	4.2%
0.10	3.75	3.87	4.14	9.5%	6.5%
0.20	5.11	5.35	5.88	13.1%	9.0%
0.40	8.38	8.86	9.93	15.6%	10.7%
0.60	13.47	14.28	15.99	15.8%	10.7%

Fig. 9. Relationship between $\overline{T_{end}}$ and loss rates when $Buf_S = 5$, $t_{pack} = 1$ and $t_{delay} = 2.5$.

3.4.3. End-to-end packet delay: T_{end_a}

T_{end_a} is the expected time for packet a to reach to User Receiver once it is given to POC Sender for transmission. In general, there are two parts of this target value: (1) the expected time to reach to POC Receiver, and (2) the expected buffering time (i.e, Buf_a). We computed (2) in Section 3.3.3. By just computing the first term, we can derive the expression for T_{end_a}.

$$E(\text{time to reach to POC Receiver}) = t_{delay} + \sum_{i=1}^{\infty}(i-1) * t_{out} * (1-p) * p^{i-1} = t_{delay} + \frac{p}{1-p} * t_{out}.$$

Hence, T_{end_a} is:

$$T_{end_a} = \left(t_{delay} + \frac{p}{1-p} * t_{out}\right) + Buf_a. \tag{9}$$

Fig. 9 illustrates the average of packet delays (i.e., $\overline{T_{end}} = (\sum_{i=1}^{N} T_{end_i})/N$) for the three *POs* in Fig. 5. For example, for the *PO* with 0.6 base density, when $Buf_S = 5$, $t_{pack} = 1$, $t_{delay} = 2.5$ and $p = q = 0.4$, the average end-to-end packet delay is 8.86 time units. This is 10.7% better than the packet delay achieved for a chain (i.e., 1.0 density). Fig. 9 shows that, at small loss rates (e.g., $p, q < 0.05$), $\overline{T_{end}}$ for all *POs* are almost identical. On the other hand, as the loss rate increases, both percentage and absolute improvements in $\overline{T_{end}}$ increase. Such improvements of partially ordered services are particularly higher with lower density *POs*.

It is noteworthy that partially ordered services provide smaller improvements in delay than those in other target values such as buffer utilization, and buffering probabilities and times. Intuitively, this is because the dominant factors in end-to-end packet delay such as network layer delays and retransmissions due to packet losses cannot be eliminated by reducing the delivery precedence constraints among packets. This intuition can easily be verified by expression (9). Consider the two parts of this expression: (1) expected time to reach to POC Sender and (2) Buf_a. The first term "$t_{delay} + \frac{p}{1-p} * t_{out}$" represents the T_{end_a} components due to network layer delays and retransmissions. This term is independent of the *PO* being used; it cannot be reduced by using a partially ordered service. Partially ordered service can improve delay only by reducing the buffering times (i.e., just one of the two important parts of delay expression). Therefore, the overall improvements in delay are not as significant as those of other target values. Nevertheless, there is still some improvement obtainable in $\overline{T_{end}}$ by partially ordered services.

In general, Section 3.4 shows that for applications that can tolerate some reordering in the delivery of packets, use of partially ordered service instead of ordered one provides some delay and considerable buffer utilization

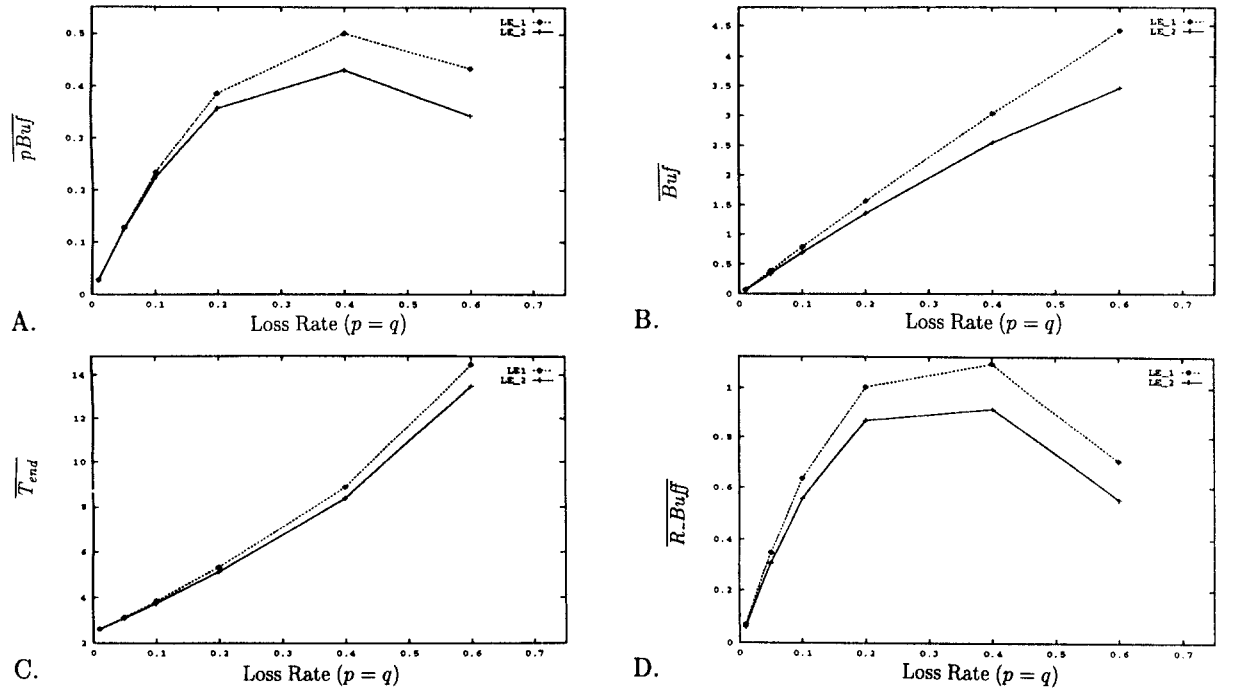


Fig. 10. Comparison of two *LE*s as transmission order when $Buf_S = 5$, $t_{pack} = 1$ and $t_{delay} = 2.5$.

improvements in the system, particularly as the loss rate increases and the order requirements of the applications decrease. Analytic results also show that in terms of throughput, it does not make any difference which service (i.e., ordered, partially ordered, unordered) an application uses.

3.5. Using a different LE as POC sender's transmission order

In a partially ordered service, POC Sender is permitted to transmit packets in any order that does not violate the partial order [3]. That is, any valid *LE* of *PO* is permitted. Does the choice of which *LE* is used by POC Sender affect the expected performance? In this section, we will address this question by comparing the performance of different *LE*s of a *PO*.

Consider the following two *LE*s of the first $PO = (a \prec c \prec e)||(b \prec d)$ in Fig. 5: $LE_1 = acebd$ and $LE_2 = abcde$. Is either of these two *LE*s a better transmission order? From Section 3.3.1, we know that if the separations of the dependent packets in transmission order (i.e, $Dist_{a,b}$ values for all $a \prec b$) are increased, then buffering probabilities and times in the system are reduced. Thus, we can say that LE_2 is better because in LE_1, dependent packets *ace* and *bd* are transmitted in sequence whereas in LE_2, they are separated.

Which target values can be improved by using LE_2 over LE_1? Since λ is independent of *PO* being used, throughput will be unaffected by the choice of *LE*. On the other hand, all other target values (i.e., $\overline{pBuf}$, $\overline{Buf}$, $\overline{T_{end}}$ and $\overline{R\_Buff}$) will be improved by LE_2.[15] This can easily be seen in Fig. 10 that compares the performance of LE_1 with that of LE_2. This figure shows that at higher loss rates, buffering probabilities and times, buffer utilization and delay are improved by using LE_2 over LE_1. For example, when the loss rate equals 0.1, the percentage improvements are 4.30%, 12.27%, 2.51% and 12.27% in $\overline{pBuf}$, $\overline{Buf}$, $\overline{T_{end}}$ and $\overline{R\_Buff}$, respectively.

[15] See expressions (5), (6), (8) and (9) for $pBuf_a$, Buf_a, $\overline{R\_Buff}$ and T_{end_a}, respectively; all these target values decrease with increasing $Dist_{b,a}$ values for $b \prec a$.

Fig. 10 clearly shows that the choice of which *LE* is used by POC Sender affects the overall system performance. Thus, it is important to use a good linear extension as transmission order in a partially ordered service.

Reference [9] provides a more in-depth study of the problem of determining the best transmission order for a given partially ordered service. In [9], a new metric (*pBuf*-metric) for quantifying a linear extension's goodness is defined based on the average of buffering probabilities (i.e., *pBuf*-metric= $(\sum_{a \prec b \text{ in } PO}(pBuf_{a,b}))/N$). This *pBuf*-metric approximates the average of buffering probabilities when the linear extension *LE* of *PO* is used as transmission order. Because of this characteristic, *pBuf*-metric can be used to discover the *LE* (or the set of *LE*s) of a *PO* that achieves the lowest buffering probabilities in the system. In deciding between two linear extensions of a *PO*, if POC Sender chooses the one with smaller *pBuf*-metric value, then that *LE* is expected to result in smaller buffering probabilities and other performance advantages.

Consistent with analytic results, the simulation results of [9] show that by choosing better *LE*s over suboptimal ones, we obtain some delay and important buffer utilization improvements, but, in general, no throughput improvement. Results of [9] also show that *pBuf*-metric (the metric derived directly from analytic model) is effective in determining the good *LE*s of a *PO* even under variable network layer delays and finite receiver buffers.

4. Verification of analytic model

Section 3 investigates the buffering, delay, buffer utilization and throughput characteristics of partially ordered transport services by computing the set of target values defined at Tables 3 and 4. In this section, we will verify the analytic model by comparing the results against those of simulation model. More specifically, the computations of the following target values will be verified: buffering probabilities and times between dependent pairs (i.e., $pBuf_{a,b}$ and $Buf_{a,b}$ for $a \prec b$), buffering probabilities and times for a packet (i.e., $pBuf_a$ and Buf_a), throughput (i.e., λ), expected buffers used at receiver (i.e., $\overline{R\_Buff}$) and end-to-end packet delay (i.e., T_{end_a}).

For simulation study of partially ordered transport services, we built an OPNET-based simulation model at the University of Delaware's Protocol Engineering Lab. OPNET (OPtimize Network Engineering Tools) is a comprehensive engineering system capable of simulating large communication networks with detailed protocol modeling and performance analysis [2].

For the verification process of analytic model, we run three different sets of experiments each of which testing a different hypothesis of the analytic model. It is important to state that these four hypotheses were all developed based only on the analytic model and before any simulation were run. The hypotheses are:

Hypothesis 1. Analytic model gives the exact expressions for $pBuf_{a,b}$, $Buf_{a,b}$ and λ.

Hypothesis 2. Analytic model results in the actual values for $pBuf_a$, Buf_a, T_{end_a} and $\overline{R\_Buff}$ when $Buf_S = 2$.

Hypothesis 3. As Buf_S increases, analytic model overestimates the values for $pBuf_a$, Buf_a, T_{end_a} and $\overline{R\_Buff}$.

Hypothesis 4. At low loss rates, the analytic model better approximates the values for $pBuf_a$, Buf_a, $\overline{R\_Buff}$ and T_{end_a} even at larger Buf_S values.

The analytic model gives the exact computational results for the target values $pBuf_{a,b}$, $Buf_{a,b}$ and λ regardless of buffer sizes and loss rates. Thus, under the assumptions given at Table 2, we have the exact expressions for these target values. In all experiments performed, we show that the simulation and analytic results are closely matched for these three target values (Hypothesis 1).

We will explain Hypotheses 2–4 only for buffering probabilities (i.e., $pBuf_a$). But one can easily extend these explanations for buffering times (i.e., Buf_a). Since T_{end_a} and $\overline{R\_Buff}$ are derived using Buf_a, the same arguments will be valid for those two target values as well.

The buffering probability for packet a is approximated in expression (5). Let *Error* be the following:

$$Error = \sum_{\forall\{b1,b2\}\prec a} P(a \text{ overtakes } b1, b2) - \sum_{\forall\{b1,b2,b3\}\prec a} P(a \text{ overtakes } b1, b2, b3) + \cdots$$
$$= P(a \text{ overtakes two or more dependent packets}) \quad (10)$$

Thus, $pBuf_a - Error$ is the exact computation where *Error* represents the error value in the approximated computation.

When $Buf_S = 2$, $P(a$ overtakes more than one preceding packet$) = 0$. Thus, under this condition, $Error = 0$ and we have the exact computation for the target values $pBuf_a$, Buf_a, T_{end_a} and $\overline{R\_Buff}$. In Experiment 1, we show that when $Buf_S = 2$, the simulation and analytic results for these target values are closely matched (Hypothesis 2).

As Buf_S increases, we expect to see that packet a overtakes more preceding packets. Thus, with increasing Buf_S, *Error* should also increase. Notice that as *Error* increases, the analytic model starts increasingly overestimating the approximated values. This effect is shown in Experiment 2 (Hypothesis 3).

Finally, at small loss rates, $Error = P(a$ overtakes more than one dependent packets) will be low even when Buf_S is large. Intuitively, this is because, at small loss rates, it is unlikely that packet a will overtake two or more preceding packets. Thus, at small loss rates, the analytic model should provide closer approximations for the target values $pBuf_a$, Buf_a, T_{end_a} and $\overline{R\_Buff}$. This is shown in Experiment 3 (Hypothesis 4).

The analytic model computes the target values under certain assumptions (e.g., constant network layer delays, infinite buffers at POC Receiver, etc.; see Table 2 for the full set of assumptions). The parameters of the simulation model are tuned so that we have a comparable system. That is, the results of these simulation experiments can be compared to analytic results because they are both derived from the same kind of system.

In the simulation study, each experiment is repeated three times with 30000 packets and the averages of the observed values from these three simulation runs are computed. The worst case for the analytic model to estimate the values for $pBuf_a$, Buf_a, T_{end_a} and $\overline{R\_Buff}$ occurs when the *PO* under consideration has the most dependencies possible. This is because for such partial orders, there will be a higher chance that the packets overtaken by a will have a dependency relation with a, and thus, $Error = P(a$ overtakes two or more dependent packets) will be higher. Additionally, if the partial order has the most dependencies possible, then we will have the maximal number of nonzero terms for $pBuf_{a,b}$ and $Buf_{a,b}$ to compare with simulation results.[16] Hence, to best verify the analytic model, in all experiments, we use a chain as *PO*.

Sections 4.1, 4.2, and 4.3 present the results of the experiments through graphs. The graphs compare the average values from the three simulation runs with the analytic values for all target values. In the graphs, the following conventions are used: the analytic results are plotted as lines without explicitly marking the computed points whereas the simulation results are shown only as computed points. Additionally, the simulation and analytic results in the graphs are labeled as "S" and "A", respectively.

4.1. Experiment 1

In these experiments, we simulated a sender buffer size (Buf_S) of 2 in order to test Hypothesis 2. The results of these experiments are also checked for Hypothesis 1. These experiments consist of three simulations runs for the loss rates of 0.01, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, and 0.8. The corresponding analytic values are computed for $p = q$ values of 0.01, 0.05, 0.1, 0.15, 0.2, 0.25, 0.3, 0.35, 0.4, 0.45, 0.5, 0.6, 0.7, 0.8, 0.9, and 0.99. The

[16] $pBuf_{a,b}$ and $Buf_{a,b}$ are zero if there is no constraint $a \prec b$ in *PO*; see Section 3.3.1.

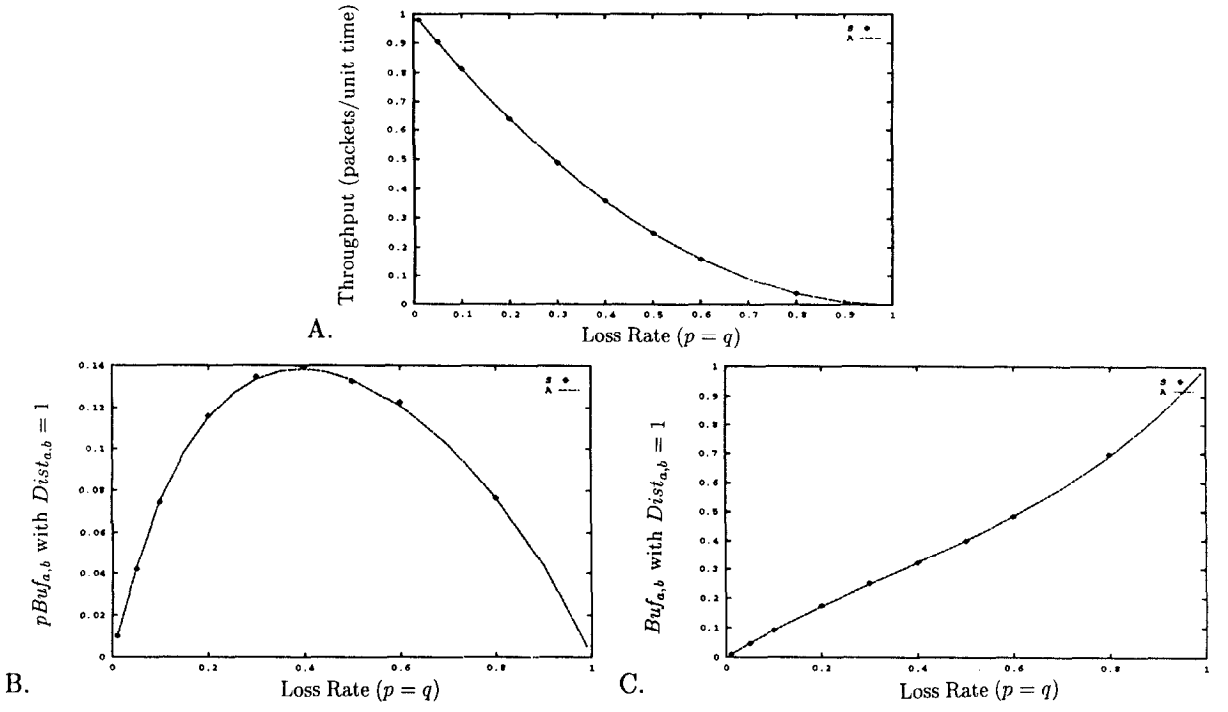


Fig. 11. Graphs for (A) λ, (B) $pBuf_{a,b}$ and (C) $Buf_{a,b}$ (Experiment 1).

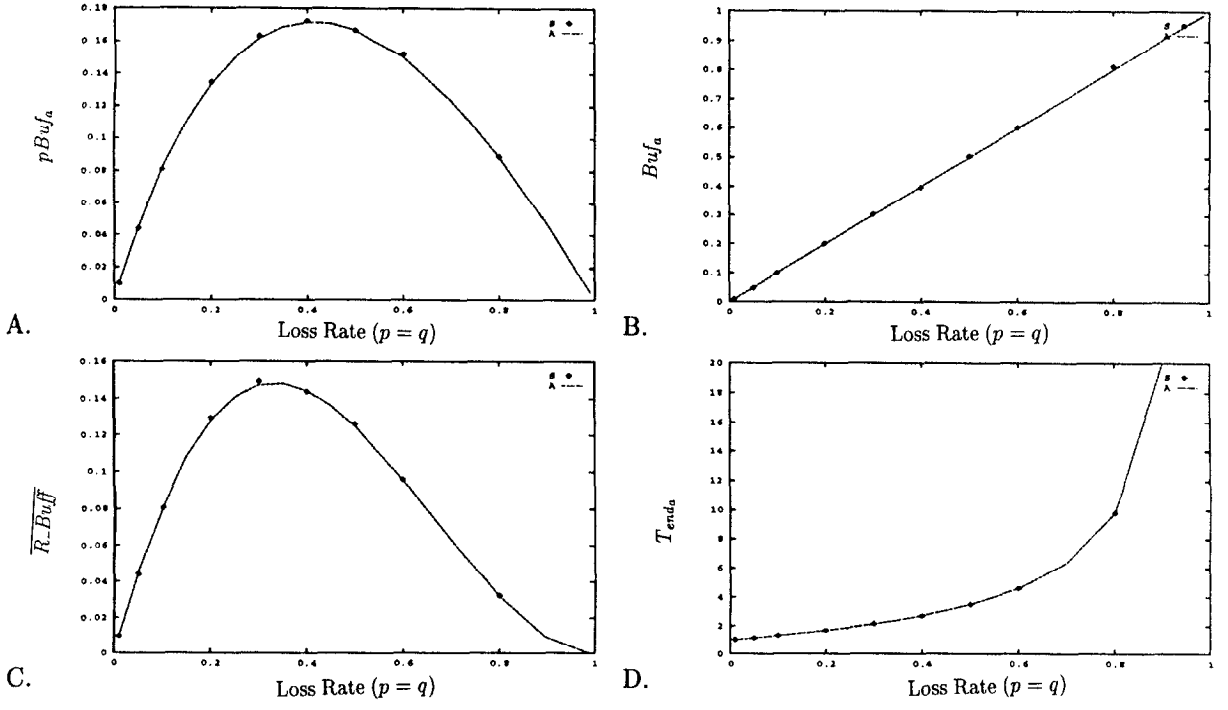


Fig. 12. Graphs for (A) $pBuf_a$, (B) Buf_a, (C) $\overline{R\_Buff}$ and (D) T_{end_a} (Experiment 1).

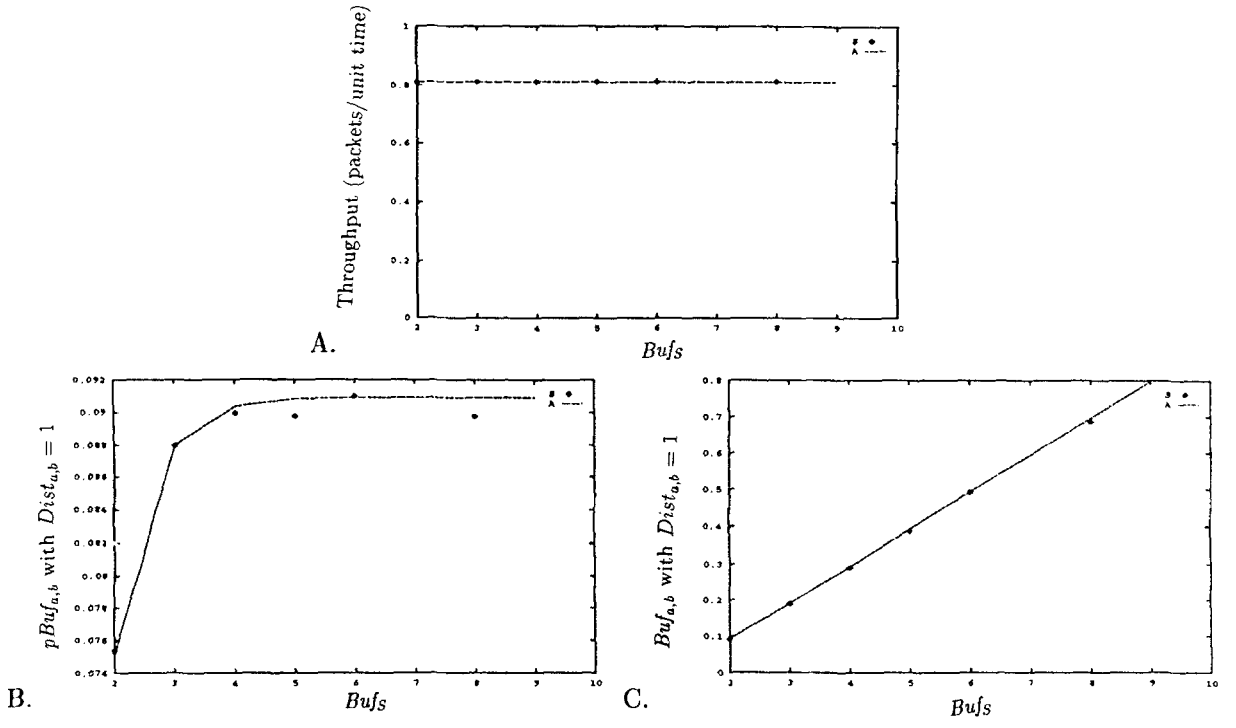


Fig. 13. Graphs for (A) λ, (B) $pBuf_{a,b}$ and (C) $Buf_{a,b}$ (Experiment 2).

analytic model computations are done for more points (i.e., loss rates) in order to have smooth curves in the graphs.

Figs. 11(A)–(C) illustrate the graphs for λ, $pBuf_{a,b}$ and $Buf_{a,b}$. As the graphs show, the analytic and simulation values are closely matched in these experiments. In general, all results from two models are within 1% of each other. Thus, the first set of experiments supports the correctness of Hypothesis 1.

The simulation and analytic values for $pBuf_a$, Buf_a, $\overline{R\_Buff}$ and T_{end_a} are given in Figs. 12(A)–(D). Since $Buf_S = 2$, the analytic and simulation results strongly support each other for these target values. The values from two models are generally within 1% of each other. Thus, as predicted by Hypothesis 2, the analytic and simulation results are essentially identical when $Buf_S = 2$.

4.2. Experiment 2

In the first set of experiments, we set $Buf_S = 2$, vary the loss rate, and test Hypothesis 1 and 2. In the second set of experiments, we set the loss rate= 0.1, vary Buf_S, and test Hypothesis 1 and 3. We run three simulations for Buf_S values of 2, 3, 4, 5, 6, and 8. The corresponding analytic values are computed for Buf_S values of 2, 3, 4, 5, 6, 7, 8, and 9.

Figs. 13(A)–(C) introduce the graphs for λ, $pBuf_{a,b}$ and $Buf_{a,b}$. As in the first set of experiments, the analytic and simulation values are close to each other in these experiments as well. Thus, the λ, $pBuf_{a,b}$ and $Buf_{a,b}$ results of these experiments are as stated in Hypothesis 1.

The graphs for $pBuf_a$, Buf_a, $\overline{R\_Buff}$ and T_{end_a} are given in Figs. 14(A)–(D). These graphs clearly show that as Buf_S increases, the analytic model increasingly overestimates these four target values as expected. For example, while at $Buf_S = 3$, the analytic model value for $pBuf_a$ is about 4.6% larger than the corresponding

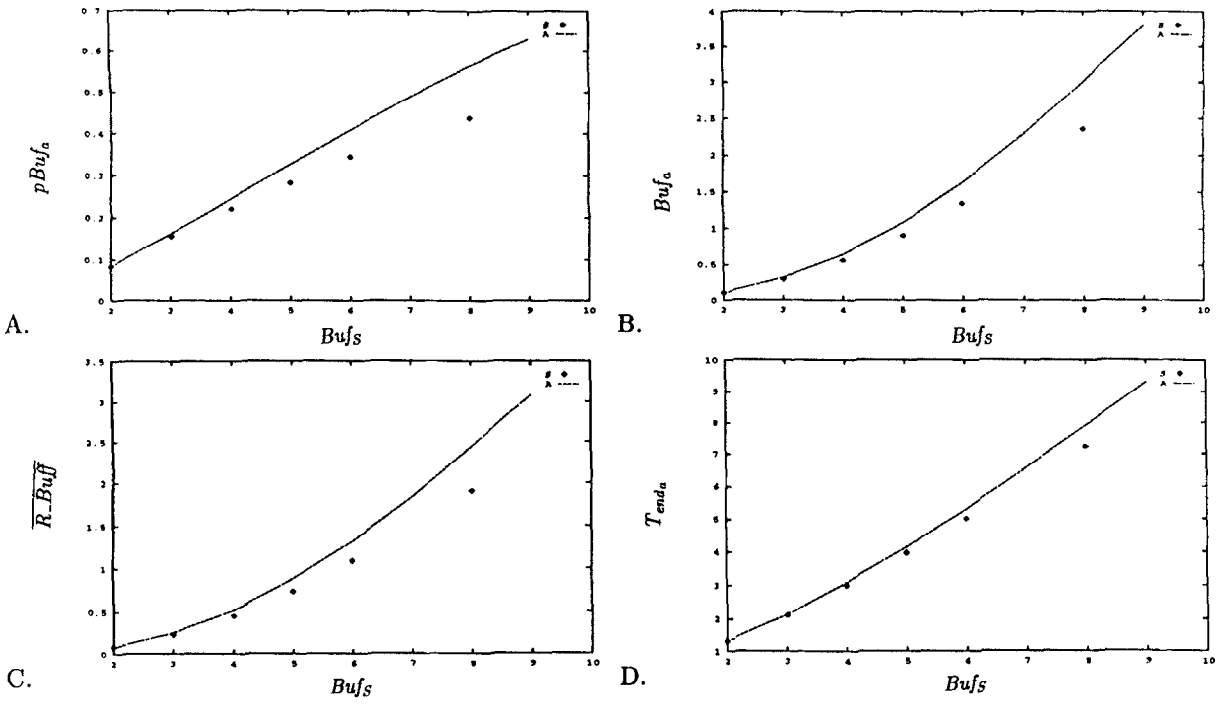


Fig. 14. Graphs for (A) $pBuf_a$, (B) Buf_a, (C) $\overline{R\_Buff}$ and (D) T_{end_a} (Experiment 2).

simulation value, at $Buf_S = 5$, it is almost 15.7% larger. Thus, Hypothesis 3 successfully explains the results of these experiments.

4.3. Experiment 3

The primary purpose of the third set of experiments is to test Hypothesis 4. Meanwhile, we also check the correctness of Hypothesis 1. In the experiments, we set Buf_S to a value greater than 2, and vary the loss rate in order to test Hypothesis 4. In both the analytic and simulation models, Buf_S is taken to be 6. The simulation experiments are run for the loss rates of 0.05, 0.1, and 0.3. The corresponding analytic values are computed for $p = q$ values of 0.05, 0.1, 0.2, and 0.3.

As stated in Hypothesis 1, Figs. 15(A)–(C) show that the analytic and simulation values for λ, $pBuf_{a,b}$ and $Buf_{a,b}$ are closely matched. The results from two models are generally within 1% of each other.

The graphs for $pBuf_a$, Buf_a, $\overline{R\_Buff}$ and T_{end_a} are illustrated in Figs. 16(A)–(D). These graphs show that the results from two models are closer to each other at lower loss rates. Thus, as predicted by Hypothesis 4, the analytic model better approximates these four target values at smaller loss rates. For example, while the analytic model overestimates T_{end_a} by about 10.9% at loss rate= 0.3, it is only 2.3% larger than the corresponding simulation value at loss rate= 0.01.

In summary, Sections 4.1–4.3 present the results from three different sets of simulation experiments. These experiments support the four hypotheses derived from the analytic model before any simulation experiment were run. In general, the analytic and simulation results are within 1% of each other for the values that analytic model is expected to provide the exact results. Based on the results of Sections 4.1–4.3, we conclude that the analytic model provides

- accurate results for λ, $pBuf_{a,b}$ and $Buf_{a,b}$ under any loss rate and sender buffer size,
- accurate results for $pBuf_a$, Buf_a, $\overline{R\_Buff}$, and T_{end_a} when $Buf_S = 2$,

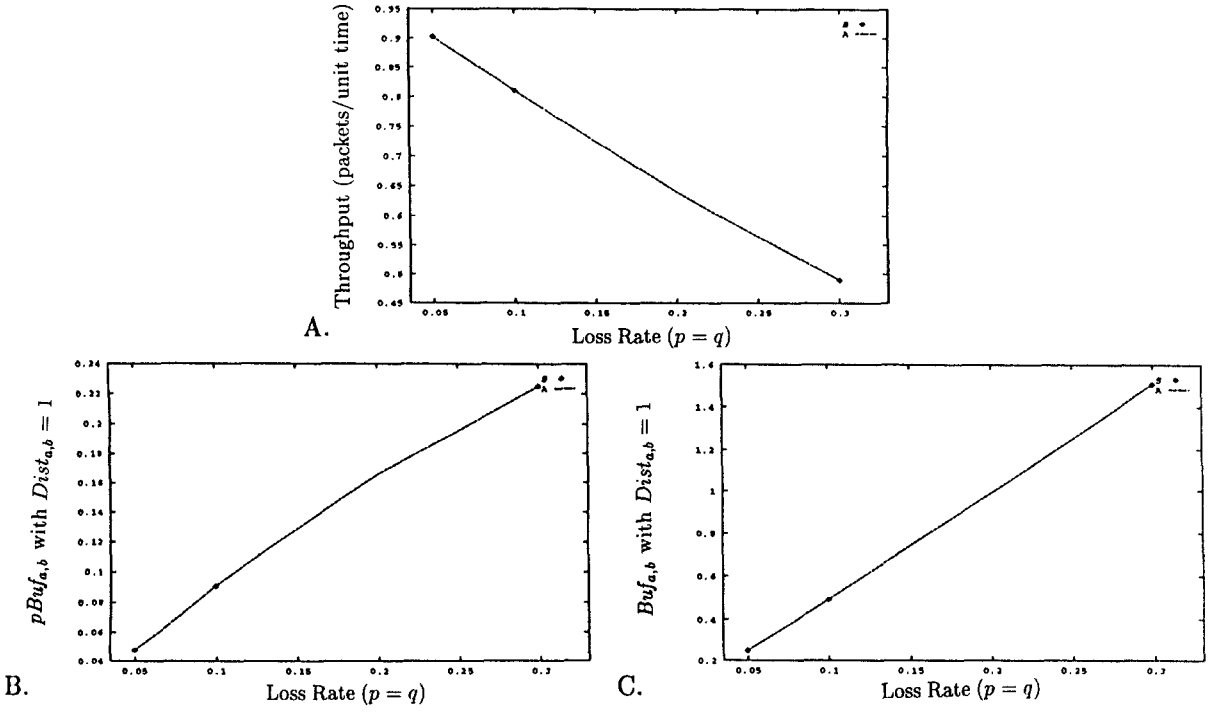


Fig. 15. Graphs for (A) λ, (B) $pBuf_{a,b}$ and (C) $Buf_{a,b}$ (Experiment 3).

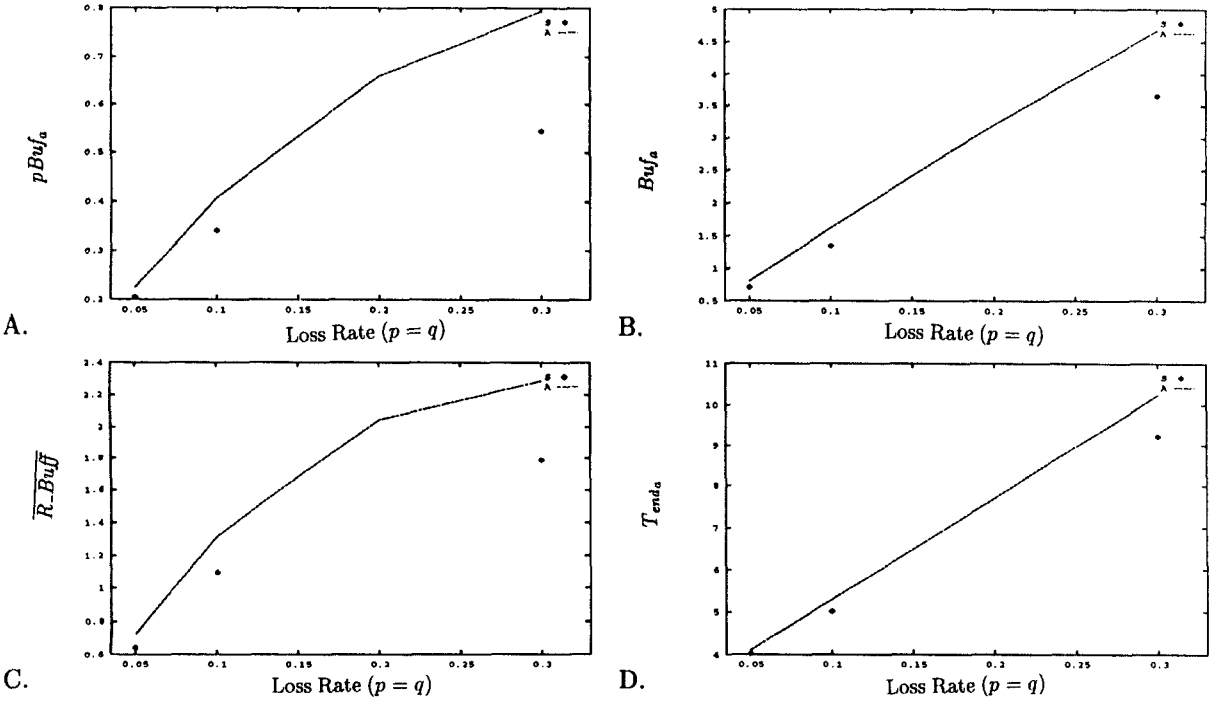


Fig. 16. Graphs for (A) $pBuf_a$, (B) Buf_a, (C) $\overline{R\_Buff}$ and (D) T_{end_a} (Experiment 3)

- close results for $pBuf_a$, Buf_a, $\overline{R\_Buff}$, and T_{end_a} when sender buffer size or the loss rate is small, and
- accurate "shape of curve" for $pBuf_a$, Buf_a, $\overline{R\_Buff}$, and T_{end_a} even at high loss rates and large Buf_S values.

Thus, based on these results, we conclude that the simulation experiments provide strong evidence to the correctness of the analytic model.

5. Summary of main results

This paper presents an analytic model for investigating the throughput, delay and buffer utilization characteristics of partially ordered transport services. Through this model, we study the effects of packet and ack losses as well as various levels of applications' order requirements on the performance of different services (i.e., ordered, partially ordered, and unordered). The analytic model is verified by comparing the results against those of an OPNET simulation model.

The analytic study shows that in terms of throughput, it does not make any difference which service (i.e., ordered, partially ordered) an application uses. On the other hand, for applications that can tolerate some reordering in the delivery of objects, use of ordered service instead of partially ordered one results in important buffer utilization and delay increases, particularly as the underlying network's loss rate increases and the applications' order requirements decrease. Unordered service, however, is unable to provide the minimal order guarantees of applications. Thus, in lossy environments, partially ordered service is necessary to provide the order requirements of applications, and at the same time, to prevent the delay and buffer utilization costs of ordered service.

In a partially ordered service, the sender is permitted to transmit packets in any order that does not violate the partial order. Analytic results show that by judicious choice of transmission order, the system performance can further be improved. Thus, it is important to use a good linear extension as transmission order in a partially ordered service.

Appendix A. Computational details

In this appendix, we present the computational details of $pSB_{a,b}$, $pBuf_{a,b}$ and $Buf_{a,b}$. $pSB_{a,b}$ is the probability that all transmissions of packet a prior to packet b's first transmission fail. This probability is the cornerstone of our computations. Obviously, only if packet a keeps failing until packet b's first transmission, is there a possibility that packet b will overtake packet a, resulting in packet b's buffering at POC Receiver. $pSB_{a,b}$ is used to derive the expressions for $pBuf_{a,b}$ and $Buf_{a,b}$, which in turn are used in the approximate computations of $pBuf_a$ and Buf_a. The expressions for $\overline{R\_Buff}$ and T_{end_a} are derived by using Buf_a. Because of this, we will present a detailed discussion on $pSB_{a,b}$ in the next section.

Let s_{a_i} be the time that packet a's i^{th} transmission starts at POC Sender. s_{a_1} is the time of a's original transmission, s_{a_2} is the time of the first retransmission, and so on. Additionally, let r_a be the time that packet a is received at POC Receiver for the first time. Throughout this appendix, we will use these two variables in our computations.

A.1. Probability that one packet keeps failing until the first transmission of another packet: $pSB_{a,b}$

Fig. A.1 shows a scenario where packet a's original transmission and all of its retransmissions occurring before packet b's first transmission fail. In this figure, the total number of times that a fails before b's first transmission is $t + 1$.

Note that for any original packet transmission, there must be a corresponding successful packet-ack transmission t_{out} earlier in time in order to free a buffer space at POC Sender and to allow for the corresponding

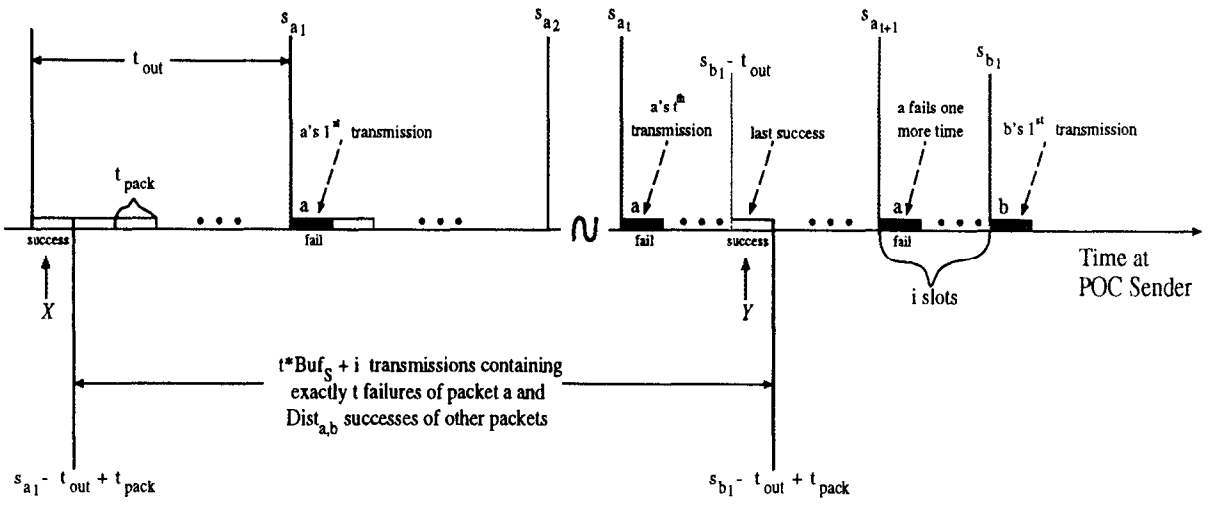


Fig. A.1. a fails until s_{b_1}

packet's first transmission. For example, the successful packet-ack transmission allowing for packet a's first transmission occurs at point X in Fig. A.1.

In this example case, $s_{b_1} - s_{a_1} = t * t_{out} + i * t_{pack} = (t * Buf_S + i) * t_{pack}$. Thus, there are a total of $t * Buf_S + i$ transmissions during time $[s_{a_1}, s_{b_1}]$ or equivalently during time $[s_{a_1} - t_{out} + t_{pack}, s_{b_1} - t_{out} + t_{pack}]$. If a fails until the first transmission of b, then the following conditions are satisfied for the $t * Buf_S + i$ transmissions during time $[s_{a_1} - t_{out} + t_{pack}, s_{b_1} - t_{out} + t_{pack}]$:

(1) a fails t times. Thus, out of $t * Buf_S + i$ transmissions, t of them are a's unsuccessful packet transmissions; leaving $t * (Buf_S - 1) + i$ transmissions for other packets.

(2) Out of remaining $t * (Buf_S - 1) + i$ transmissions, there must be only $Dist_{a,b}$ successful packet-ack transmissions allowing for the packets between a and b (a total of $Dist_{a,b}$ of them) to have their first transmissions in time $[s_{a_1} + t_{pack}, s_{b_1} + t_{pack}]$.

(3) A successful packet-ack transmission occurs precisely at time $s_{b_1} - t_{out}$ (point Y in Fig. A.1), thereby allowing POC Sender to transmit packet b at time s_{b_1}. This is the last success.

(4) The remaining $Dist_{a,b} - 1$ successes must have occurred in any of the previous $t * (Buf_S - 1) + i - 1$ transmissions.

Additionally, a must fail one more time during $[s_{b_1} - t_{out} + t_{pack}, s_{b_1}]$ just before b's first transmission as shown in Fig. A.1. In this example scenario, there are t timeout periods between the first transmissions of packets a and b, and b's first transmission takes place at the $(i+1)$th slot of the last timeout period (i.e., $(t+1)$th timeout period after a's first transmission). We have the following overall probability for this situation:

$$
\begin{aligned}
&P(a \text{ fails until } s_{b_1} \text{ and there are } t \text{ timeout periods in } [s_{a_1}, s_{b_1}] \\
&\quad \text{and } b\text{'s first transmission takes place at } (i+1)\text{th slot of last timeout period}) \\
&= P(a \text{ fails } t \text{ times in } [s_{a_1} - t_{out} + t_{pack}, s_{b_1} - t_{out} + t_{pack}] \text{ and there is a success at time } s_{b_1} - t_{out} \\
&\quad \text{and there are } Dist_{a,b} - 1 \text{ successes from remaining } t * (Buf_S - 1) + i - 1 \text{transmissions} \\
&\quad \text{and } a \text{ fails one more time in time } [s_{b_1} - t_{out} + t_{pack}, s_{b_1}]) \\
&= p^t * (p_{succ}) * \left(\binom{t * (Buf_S - 1) + i - 1}{Dist_{a,b} - 1} (p_{succ})^{Dist_{a,b}-1} * (1 - p_{succ})^{t*(Buf_S-1)+i-Dist_{a,b}} \right) * p
\end{aligned}
$$

$$= p^{t+1} * \left(\binom{t * (Buf_S - 1) + i - 1}{Dist_{a,b} - 1} (p_{succ})^{Dist_{a,b}} * (1 - p_{succ})^{t*(Buf_S-1)+i-Dist_{a,b}} \right). \quad (A.1)$$

Since $1 \leqslant i \leqslant Buf_S - 1$:

$$P(a \text{ fails until } s_{b_1} \text{ and there are } t \text{ timeout periods in } [s_{a_1}, s_{b_1}])$$

$$= p^{t+1} * \left(\sum_{i=1}^{Buf_S-1} \binom{t * (Buf_S - 1) + i - 1}{Dist_{a,b} - 1} (p_{succ})^{Dist_{a,b}} * (1 - p_{succ})^{t*(Buf_S-1)+i-Dist_{a,b}} \right) \quad (A.2)$$

Based on these observations, $pSB_{a,b}$ can be computed as follows:

$$pSB_{a,b} = P(a \text{ fails until } s_{b_1})$$

$$= \sum_{t=\lfloor (Dist_{a,b}-1)/Buf_S-1 \rfloor}^{\infty} P(\text{there are } t \text{ timeout periods in } [s_{a_1}, s_{b_1}] \text{ and } a \text{ fails until } s_{b_1})$$

$$= p^{\lfloor (Dist_{a,b}-1)/Buf_S-1 \rfloor+1}$$

$$* \sum_{i=\text{remainder}((Dist_{a,b}-1)/(Buf_S-1))+1}^{Buf_S-1} \binom{\lfloor (Dist_{a,b} - 1)/(Buf_S - 1) \rfloor * (Buf_S - 1) + i - 1}{Dist_{a,b} - 1}$$

$$* \left((p_{succ})^{Dist_{a,b}} * (1 - p_{succ})^{\lfloor (Dist_{a,b}-1)/(Buf_S-1) \rfloor * (Buf_S-1)+i-Dist_{a,b}} \right)$$

$$+ \sum_{t=\lfloor (Dist_{a,b}-1)/(Buf_S-1) \rfloor+1}^{\infty} p^{t+1} \sum_{i=1}^{Buf_S-1} \binom{t * (Buf_S - 1) + i - 1}{Dist_{a,b} - 1}$$

$$* \left((p_{succ})^{Dist_{a,b}} * (1 - p_{succ})^{t*(Buf_S-1)+i-Dist_{a,b}} \right) \quad (A.3)$$

Expression (A.3) thus far has no known closed-form solution; it reduces to expression (A.4) when $Buf_S = 2$.

$$pSB_{a,b} = p^{Dist_{a,b}} * (p_{succ})^{Dist_{a,b}} + \sum_{t=Dist_{a,b}}^{\infty} \binom{t}{Dist_{a,b} - 1} p^{t+1} * (p_{succ})^{Dist_{a,b}} * (1 - p_{succ})^{t+1-Dist_{a,b}}$$

$$= \left(\frac{p * p_{succ}}{1 - p + p * p_{succ}} \right)^{Dist_{a,b}} = \left(\frac{p * (1 - q)}{1 + p * (1 - q)} \right)^{Dist_{a,b}} \quad (A.4)$$

A.2. Computation of $pBuf_{a,b}$

$pBuf_{a,b}$ is the probability that packet a is 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.)

$$pBuf_{a,b} = P(r_a > r_b) = pSB_{a,b} * P(r_a > r_b \mid a \text{ fails until } s_{b_1}) \quad (A.5)$$

The latter term in expression (A.5) can be computed as follows:

$$P(r_a > r_b \mid a \text{ fails until } s_{b_1})$$

$$= \sum_{j=1}^{\infty} (1 - p) * p^{j-1} * P(r_a > r_b \mid b\text{'s packet transmission succeeds at } j\text{th attempt and } a \text{ fails until } s_{b_1})$$

$$= \sum_{j=1}^{\infty} (1-p) * p^{j-1} * \left(\sum_{i=j}^{\infty} P(a\text{'s packet transmission succeeds at } i\text{th attempt after } s_{b_1} \text{ where } i \geqslant j) \right)$$

$$= \sum_{j=1}^{\infty} (1-p) * p^{j-1} * \left(\sum_{i=j}^{\infty} (1-p) * p^{i-1} \right)$$

$$= \frac{1}{1+p}. \tag{A.6}$$

Substituting equation (A.6) into expression (A.5):

$$pBuf_{a,b} = pSB_{a,b} * \frac{1}{1+p} \tag{A.7}$$

A.3. Computation of $Buf_{a,b}$

The value $Buf_{a,b}$, the time that packet b is buffered waiting for packet a to arrive, is computed for $a \prec b$. This value is zero by definition when there is no dependency between a and b since in this case b never has to wait for a in the buffers of POC Receiver.

$$\begin{aligned}
Buf_{a,b} &= E(r_b - r_a) \\
&= pSB_{a,b} * E(r_b - r_a \mid a \text{ fails until } s_{b_1}) \\
&= pSB_{a,b} * \left(\sum_{j=1}^{\infty} (1-p) * p^{j-1} \right. \\
&\quad \left. * E(r_b - r_a \mid a \text{ fails until } s_{b_1} \text{and } b\text{'s packet transmission succeeds at } j\text{th attempt}) \right) \\
&= pSB_{a,b} * \left(\sum_{j=1}^{\infty} (1-p) * p^{j-1} * \left(\sum_{i=j}^{\infty} P(a\text{'s packet transmission succeeds at } i\text{th attempt after } s_{b_1}) \right. \right. \\
&\quad \left. \left. * \left((j-i) * t_{out} + (Buf_S - SL_{a,b}) * t_{pack} \right) \right) \right) \\
&= pSB_{a,b} * \left(\sum_{j=1}^{\infty} (1-p) * p^{j-1} * \left(\sum_{i=j}^{\infty} (1-p) * p^{i-1} * \left((j-i) * t_{out} + (Buf_S - SL_{a,b}) * t_{pack} \right) \right) \right) \\
&= pSB_{a,b} * \frac{(1-p) * (Buf_S - SL_{a,b}) * t_{pack} + p * t_{out}}{1-p^2} \\
&= pBuf_{a,b} * \frac{(1-p) * (Buf_S - SL_{a,b}) * t_{pack} + p * t_{out}}{1-p}
\end{aligned} \tag{A.8}$$

where $SL_{a,b}$ is the expected number of slots between a's failure just before b's first transmission and b's first transmission (in Fig. A.1, this is shown as "i slots"). We have studied the events leading to a's failure until the first transmission of b in detail in Appendix A.1. Thus, by using these events, $SL_{a,b}$ can be computed as:

$$SL_{a,b} = \sum_{i=1}^{\infty} i * P(a \text{ is transmitted at 1st slot of timeout period and } b \text{ is transmitted at } (i+1)\text{th slot of the same timeout period} \mid a \text{ fails until } s_{b_1})$$

$$= \sum_{i=\text{remainder}((Dist_{a,b}-1)/(Buf_S-1))+1}^{Buf_S-1} \binom{\lfloor (Dist_{a,b}-1)/(Buf_S-1) \rfloor * (Buf_S-1)+i-1}{Dist_{a,b}-1} * i$$
$$* \left((p_{succ})^{Dist_{a,b}} * (1-p_{succ})^{\lfloor (Dist_{a,b}-1)/(Buf_S-1) \rfloor * (Buf_S-1)+i-Dist_{a,b}} \right)$$
$$+ \sum_{t=\lfloor (Dist_{a,b}-1)/(Buf_S-1) \rfloor+1}^{\infty} \sum_{i=1}^{Buf_S-1} \binom{t*(Buf_S-1)+i-1}{Dist_{a,b}-1} * i$$
$$* \left((p_{succ})^{Dist_{a,b}} * (1-p_{succ})^{t*(Buf_S-1)+i-Dist_{a,b}} \right). \quad \text{(A.9)}$$

References

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

[2] N. Baxter, H. Chien, A. Loreen, K. Marshall and S. Baraniuk, *OPNET Manual*, MIL 3, Inc, 1993.

[3] T.J. Connolly, P.D. Amer and P.T. Conrad. RFC-1693, An extension to TCP: Partial order service.

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

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

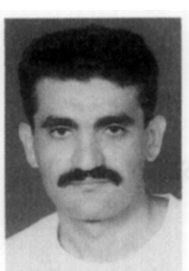
[6] M. Diaz, A. Lozes, C. Chassot and P. Amer, Partial order connections: A new concept for high speed and multimedia services and protocols, *Annals of Telecommunications* 49 (5–6) (1994) 270–281.

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

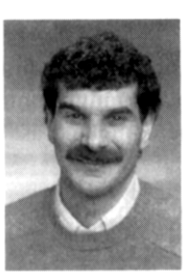
[8] R. Marasli, Partially ordered and partially reliable transport protocols: Performance analysis, Ph.D. Thesis, University of Delaware, 1997.

[9] R. Marasli, P.D. Amer and P.T. Conrad, Optimizing partially ordered transport services for multimedia applications, in: *Third International Conference on Multimedia Modeling*, Toulouse, France, Nov. 1996.

[10] R. Marasli, P.D. Amer and P.T. Conrad, Retransmission-based partially reliable transport service: An analytic model, in: *IEEE INFOCOM'96*, San Fransisco, CA, March 1996 (IEEE Press, New York, 1996) 621–629.



Rahmi Marasli received the B.S. degree in Computer Engineering from Middle East Technical University, Ankara, in 1992 and the M.S. degree in Computer and Information Sciences from University of Delaware in 1995. He is currently a Ph.D. candidate at University of Delaware. His current research interests are in protocol design and analysis, transport protocols for multimedia applications and performance analysis. He is a student member of IEEE.



Paul D. Amer received the B.S. degree *summa cum laude* in Mathematics from SUNY at Albany in 1974, and the M.S. and Ph.D. degrees in Computer and Information Science in 1976 and 1979, resp., from the Ohio State University. Since 1979, he has been at the University of Delaware where currentlt he is a professor of computer science. From 1978 to 1987, he was employed permanent part-time in Washington, DC, as a Research Computer Scientist at the National Bureau of Standards. In 1985–1986, he spent a one year at the Agence de l'Informatique in Paris contributing to the development of Estelle, now ISO International Standard 9074. In 1992–1993, he spent a one year at the Laboratoire d'Automatique et d'Analyse des Systemes (LAAS) of the Centre National de la Recherche Scientifique (CNRS) in Toulouse, France working on a partial order transport service. Currently, he is on a one year fellowship in the University of Delaware's Center for Advanced Study. His current research interests are protocols for high-speed networks, formal specifications of ISO protocols and services, protocol visualization of specifications, automatic protocol test case generation and extensions of Estelle.

Phillip T. Conrad is currently a Ph.D. candidate and visiting lecturer at the University of Delaware. He received the B.S. degree *magna cum laude* from West Virginia Wesleyan College in 1985, and the M.S. degree in computer science form West Virginia University in 1988. His research interests include the design and implementation of communications protocols and multimedia systems, as well as algorithm design and analysis.