Metrics For Quantifying Partially Ordered Transport Services *

Rahmi Marasli Paul D. Amer† Phillip T. Conrad

Computer and Information Science Department
University of Delaware, Newark, DE 19716 USA
Email: {marasli,amer,pconrad}@cis.udel.edu
Phone: (302) 831-1944 Fax: (302) 831-8458

Abstract

Partially ordered transport service offers a middle ground between ordered service and unordered service. For applications requiring only partial order rather than total order, partially ordered service provides performance improvements in terms of delay and buffer utilization. Intuitively, one expects a partially ordered service to provide relatively greater performance improvement when the partial order specified by the service user is more "flexible" (i.e., has fewer order constraints) and smaller performance improvement when the partial order is less "flexible" (i.e., has more order constraints). In this paper, we investigate this notion formally by proposing metrics for the "flexibility" of partial orders, and determining through an OPNET simulation how well these metrics correlate with expected performance. Two metrics are investigated: $m(PO)$, and *density*. Results show that for fixed system conditions (e.g., fixed buffer size, network loss rate, and round-trip delay,) $m(PO)$ and *density* correlate highly with the performance statistics investigated. However, since *density* is significantly easier to calculate, our conclusion is that *density* is the best way to rank partial orders as to their ability to provide performance improvements using partially ordered service.

Keywords: transport protocol, partially ordered service, multimedia, correlation, simulation, quality of service

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

*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.

†Corresponding author.

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 delay 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.

To achieve better tradeoffs between order and other quality-of-service (QoS) parameters (e.g., delay), and to satisfy the minimal order requirements of applications, partially ordered transport service has been proposed [1, 5, 7]. 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 delay 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* service to its users [1, 5, 7]. POC enhances an unreliable and 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. This is illustrated in Figure 1 in which reliability and order are shown as orthogonal axes. Ordered, reliable service (e.g., TCP) is represented by a single point at the upper right, and unordered, unreliable service (e.g., UDP) is represented by a single point at the lower left. POC, on the other hand, provides a range of services covering the entire plane. Analytic study has formally confirmed the intuitive results that, in general,

- a partially reliable service provides lower delay and higher throughput than a reliable service [11], and
- a partially ordered service provides lower delay than an ordered service while consuming less buffer space [10].

A question arises as to what units should be used to label the axes. If we represent unreliable service by 0 and reliable service by 1, we could say that the reliability axis is labeled with the probability of delivering a packet. In this paper, we attempt to find a similar metric for the order axis. What is desirable is a metric where ordered service corresponds to 1, unordered service corresponds to 0, and the partial orders that lie in between have meaningful values in this range. In particular, we would like the values for this metric to correlate highly with the expected performance statistics for partially ordered service.

Therefore, we investigate two metrics of a partial order's flexibility and the correlation between these metrics and the performance observed in the corresponding partially ordered service. The two metrics considered are $m(PO)$ and *density*. Reference [1] proposes $m(PO)$, a metric based on number of linear extensions of a partial order, as a complexity measure of different partially ordered services. Density [9] is another metric that measures the flexibility of a partial order.

This paper studies by way of simulation the correlation of these two metrics to four important transport layer performance statistics: throughput, average end-to-end packet delay, standard deviation of end-to-end packet delay, and average buffer utilization. Having a metric with high correlation to performance statistics would allow us (for a particular set of network conditions) to characterize the "region" of partial orders where partially ordered service offers significant performance improvements

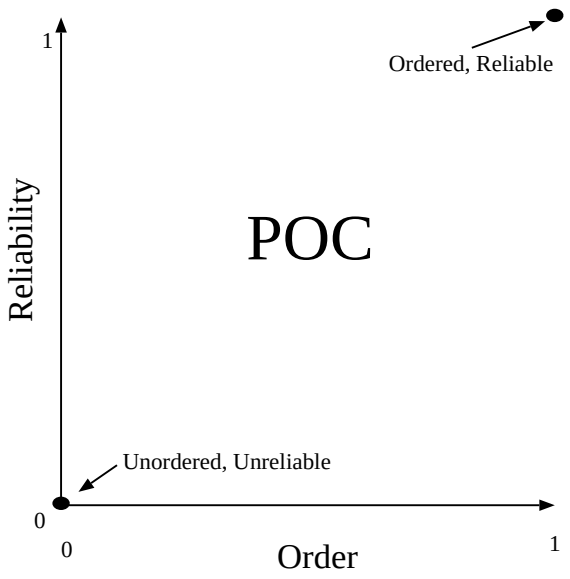

Figure 1: Reliability vs. Order

over ordered service. Our hypothesis is that either $m(PO)$ or *density* can be used for this purpose. To determine if this is true, we use simulation to generate correlation values between these two metrics and the performance statistics over a wide range of fixed independent system conditions (i.e., fixed network loss rate, fixed buffer size, and fixed network layer delay).

The paper is organized as follows: Section 2 introduces a partially ordered service and motivates it with two example applications. The metrics $m(PO)$ and density are formally defined in Section 3. Section 4 explains the experimental design for the simulation study and presents the results.

2 Why Use a Partially Ordered Service?

References [1, 7] introduce the development and motivation for a partially ordered protocol/service including several examples. A summary of this material 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 must do a "screen refresh" on a workstation screen/display containing multiple windows (see Figure 2). 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

ments 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.

3 How to Quantify Partially Ordered Services

Analytic study shows that, in general, a partially ordered service provides increasingly better performance as the precedence constraints among the objects decrease. That is, if a more flexible partial order (PO) is used, then the overall system performance can be improved. In this section, we introduce two metrics as possible candidates for quantifying partially ordered services: $m(PO)$ and *density*. Each of these metrics measures the flexibility of a partial order from a different point of view. In the simulation study of Section 4, we determine how well, if at all, these metrics correlate to the expected system performance.

3.1 A metric based on number of linear extensions: $m(PO)$

The complexity of a partial order PO can be quantified by its set of linear extensions, denoted $L(PO)$. Each linear extension in the set $L(PO)$ is essentially one of the orderings of the objects that is permitted by PO. From a purely theoretical point of view, the number of linear extensions of PO, denoted $e(PO)$, is thought as the best single number which measures the complexity of PO [14]. Clearly, for N objects, e(complete order) $= 1$ and e(no order) $= N!$ It is argued in [1] that $e(PO)$ appropriately quantifies a desired partially ordered service in communication networks. Intuitively this metric correlates to the level of effort and resources a protocol would have to use to provide a particular partial order service. This is because the larger the number of permitted orderings allowed at the receiving application (i.e., transport service user), the less overhead is expected for a protocol (i.e., transport service provider) to provide acceptable object delivery.

One of the main problems with $e(PO)$ is that it gets large very fast with increasing number of objects in PO. To avoid such large numbers, $m(PO)$ is defined in [1] as a normalized logarithmic scale of $e(PO)$. $m(PO)$ is a normalized partial order metric in the interval $[0, 1]$ where 0 represents an ordered service, values from 0 to 1 represent increasingly more flexible partially ordered service, and 1 represents unordered service:

$$m(PO) = \frac{\log e(PO)}{\log N!} \quad (1)$$

Using a metric based on $e(PO)$ presents some difficulties since computing $e(PO)$ for an arbitrary partial order is $\#P - Complete$, and it is therefore highly unlikely that any polynomial algorithm exists for this computation [3, 4].

3.2 A metric based on precedence constraints: density

The density of a PO is defined as follows [9]. Let a partial order PO be represented as a transitively closed 0-1 matrix of size N by N, where $a_{i,j} = 1$ iff $i \prec j$ in PO. (In this representation, $a_{i,i} = 0$, for all i.) Let D be defined as $D = \sum_{i=1}^{N} \sum_{j=1}^{N} a_{ij}$. D is the total number of restrictions in PO, or the number of edges in the transitively closed precedence graph. The maximum value for D is $N(N-1)/2$, therefore the density, d, is defined by the ratio $d = 2D/[N(N-1)]$ and ranges over the interval $[0, 1]$.

Density correlates intuitively with the flexibility of a partial order; a chain has a value of 1, while an antichain has a value of 0. The density also has the advantage of being relatively easy to compute.

4 Simulation Study

In Section 3, we introduce two metrics as possible candidates for quantifying partially ordered transport services. In this section, by way of simulation, we show that both of these metrics correlate highly to various performance statistics. But first, we introduce the definition of correlation and performance statistics, and the partial orders used in experiments.

4.1 Definition of Correlation

The correlation coefficient between two variables X and Y, denoted by $\rho(X, Y)$, is defined [13], as long as $Var(X) * Var(Y)$ is nonzero, by:

$$\rho(X, Y) = \frac{Cov(X, Y)}{\sqrt{Var(X) * Var(Y)}} \quad (2)$$

where $Cov(X, Y) = E[XY] - E[X]E[Y]$ is the covariance between X and Y.

The correlation coefficient is a measure of the degree of linearity between X and Y. A value of $\rho(X, Y)$ near $+1$ or -1 indicates a high degree of linearity between X and Y, whereas a value near 0 indicates lack of such linearity. Additionally, a positive value of $\rho(X, Y)$ indicates that Y tends to increase with increasing X, whereas a negative sign indicates that Y tends to decrease with increasing X.

In general, as $|\rho(X, Y)|$ gets closer to 1, we can make more accurate predictions of Y through X and vice-versa. As an example, suppose that for metrics M_1 and M_2, and QoS parameter *delay*, we have $\rho(M_1, delay) = 0.60$ and $\rho(M_2, delay) = -0.75$. Then, we can conclude that M_2 is a better metric than M_1 in predicting *delay*. Additionally, in such a case, if *delay* is the only QoS parameter that concerns us, then we can conclude that M_2 is a stronger metric than M_1 in quantifying partially ordered services.

4.2 Performance Statistics of a Partially Ordered Service

Table 1 defines four performance statistics for a partially ordered transport service. Throughput, λ, is the rate at which the transport service delivers packets to the receiving application. End-to-end packet delay, $\overline{T_{end}}$, is the average time for a packet to reach to the receiving application once it is given to the sending transport entity. For many applications such as real time audio and video, lower delay is more important than higher throughput. $STD(T_{end})$ is the standard deviation of the end-to-end packet delay. Multimedia applications generally consist of different streams such as video and

Throughput (λ)	Average number of packets delivered to receiving application per unit time
End-to-end Packet Delay ($\overline{T_{end}}$)	Average end-to-end packet delay
$STD(T_{end})$	Standard deviation of end-to-end packet delay
Receiver Buffer Utilization($\overline{R\_Buff}$)	Average number of packets buffered at receiver waiting to be delivered to application

Table 1: Definition of Performance Statistics

audio, and objects that need to be synchronized with each other. Generally, if the variation on the delays (i.e., $STD(T_{end})$) is smaller, then a finer synchronization among different streams or objects can be achieved. Hence, $STD(T_{end})$ quantifies a system's jitter. Finally, buffer utilization at the receiver, $\overline{R\_Buff}$, indicates the average memory resources that the transport protocol must allocate to satisfy an application. In general, it is desirable to have higher λ, and lower $\overline{T_{end}}$, $STD(T_{end})$ and $\overline{R\_Buff}$.

4.3 Partial Orders Used In Experiments

There exists a large number of partial orders from which to choose for our experiments. The partial orders chosen can be classified into five classes: chain-singleton, chain-of-antichain, antichain-chain, parallel-streams, and random. The first four classes are motivated by multimedia applications. Random partial orders are not suggested by any real application; we use them solely for mathematical investigation. The random POs are generated by methods discussed in [9].

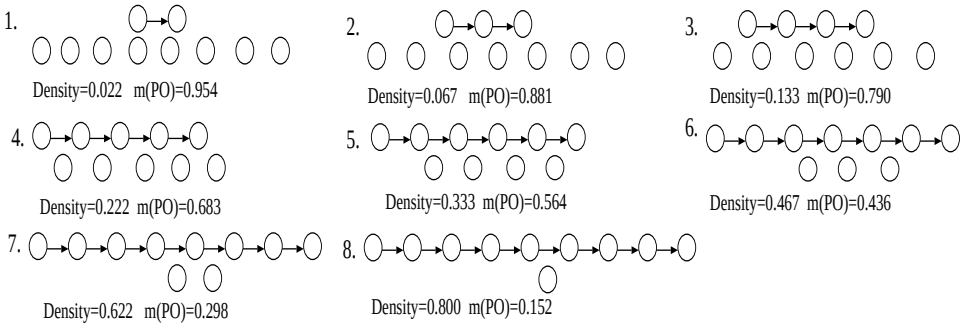

Figure 3: Chain-Singleton POs (Directed Graph Representation)

1. Chain-Singleton POs: These partial orders contain one chain and a set of singletons (see Figure 3). Such POs can be represented as two components composed in parallel: $C||S$ where $C = c_1 \prec .. \prec c_m$ and $S = s_1||..||s_l$ are the chain and the singleton components, respectively. Consider an application that opens with a welcome message and concurrently paints the screen with non-overlapping objects. The welcome message can be represented by a chain where each word (or sentence) is a separate object. The objects put on the screen as they arrive from the network can be identified as singletons (i.e., antichain). In general, any application that contains an audio or video stream in parallel with some independent objects to be displayed can be represented by this partial order class.

2. Chain-of-Antichain POs: These partial orders contain several antichains in sequence (see Figure 4). Such POs can be represented as several components composed in chain: $A_1 \oplus .. \oplus A_m$

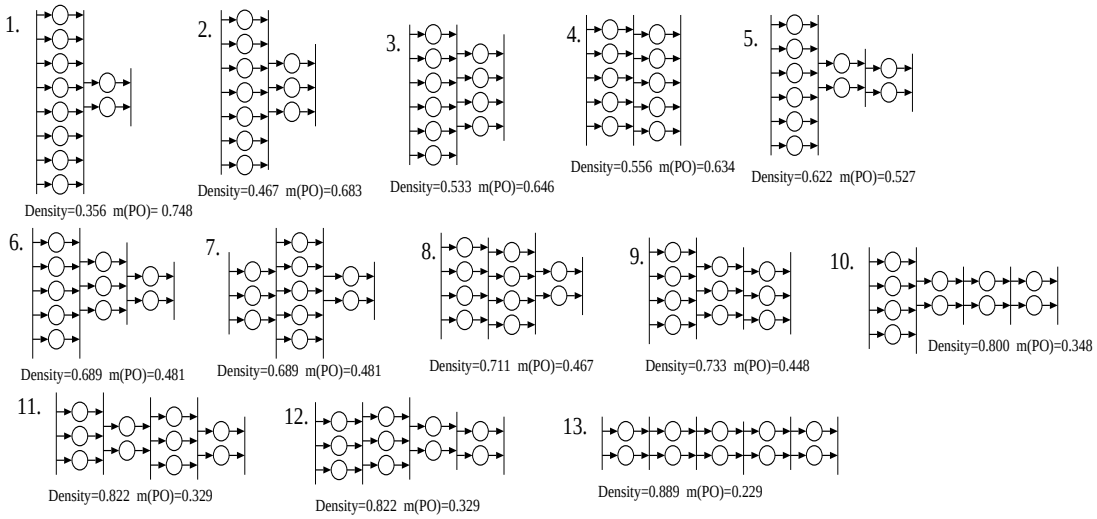

Figure 4: Chain-of-Antichain POs (Petri-Net Representation)

where[2] each component $A_i = a_1||..||a_m$ is an antichain. Consider an application that displays a screen full of non-overlapping objects, and then moves on to the next screen either after a certain amount of time or by an interaction from user. In each screen, the non-overlapping objects will be painted as they arrive. Thus, each screen in such an application can be represented by an antichain. Additionally, the objects in one screen should precede everything in the next screen. Hence, the order requirements of such applications can be represented by chain-of-antichain POs.

3. Antichain-Chain POs: These partial orders contain an antichain and a chain part in sequence (see Figure 5). Such POs can be represented as two components composed in chain: $A \oplus C$ where $A = a_1||..||a_m$ and $C = c_1 \prec .. \prec c_l$ are the antichain and chain components, respectively. Consider an application that opens with a screen containing non-overlapping objects for different icons and some buttons for an audio or video presentation. Then, based on the user input, the application starts up an audio or video. Screen that contains non-overlapping objects can be identified by an antichain. Notice that this screen should precede the up-coming audio or video presentation that can be represented by a chain. Thus, the order requirements of such applications can be identified by this partial order class.

4. Parallel-Streams POs: These are partial orders composed of multiple streams in parallel (see Figure 6). Such POs can be represented as $S_1||..||S_n$ where each $S_i = s_1 \prec .. \prec s_m$ is a stream. The applications that contain independent streams (e.g., audio, video, or subtitle streams) in parallel can be represented by this class of partial orders.

4.4 Simulation Experiments

At the University of Delaware's Protocol Engineering Lab, we built an OPNET-based simulation model to investigate the performance of partially ordered services. OPNET (OPtimize Network

[2] "$\oplus$" is the linear sum or concatenation operator for POs defined [8] as $x \prec y$ in $P \oplus Q$ if and only if $x, y \epsilon P$ and $x \prec y$ in P, or $x, y \epsilon Q$ and $x \prec y$ in Q, or $x \epsilon P$ and $y \epsilon Q$.

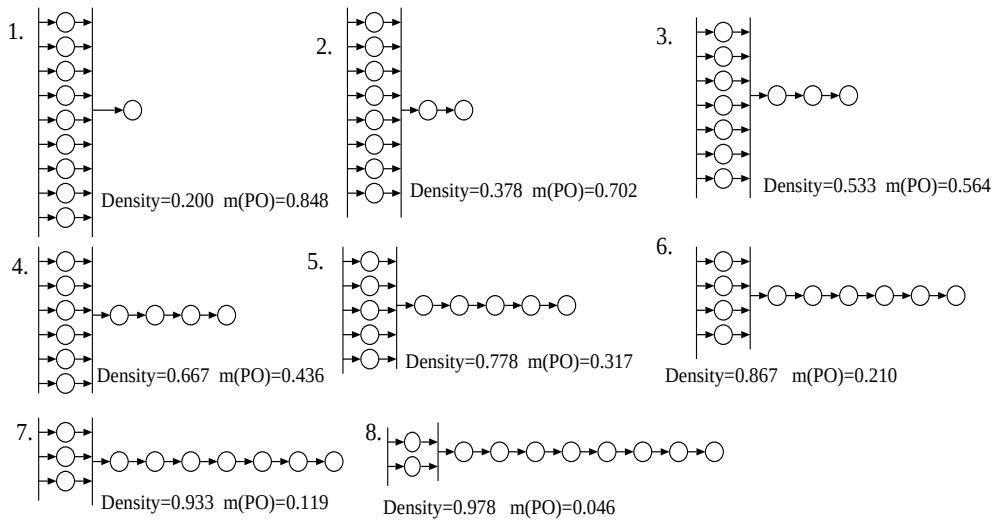

Figure 5: Antichain-Chain POs (Petri-Net Representation)

Engineering Tools) is a comprehensive engineering system capable of simulating large communication networks with detailed protocol modeling and performance analysis [2]. The simulation model was verified by

- detailed code-inspection and debugging,
- comparing the results against those of the analytic model (whenever possible), and
- designing a set of 22 experiments, stating their expected results, running the experiments, and verifying the results as expected [12].

In the simulation model's verification phase, results for λ, $\overline{R\_Buff}$, and $\overline{T_{end}}$ were generally within 1% of the analytic model results when each experiment was repeated **three** times with $30,000$ objects [12]. For the current study, each simulation experiment is repeated **five** times with $30,000$ objects, hence the results reported in this paper are expected also to be within 1% of the actual values.

There existed a large number of independent system parameters (e.g., loss rates, buffer sizes) to study in our experiments. It was impractical to exhaustively simulate millions of possible system configurations. Because of this, in our study, we only focused on three important parameters: loss rates, buffer sizes and network layer delays. The values simulated for these parameters were as follows:

- **Loss rates=** 0.01, 0.05, 0.1, 0.2, 0.4, 0.6 (6 different loss rates)
- **Buffer sizes=** Receiver:5; Sender:3, 5, 10 (3 different buffer sizes)
- **One-way network layer delays=** Normal($\mu = 4,6; \sigma = 0.25 * \mu$) (2 different delays)

We simulated loss levels ranging from 1% up to 60% which is well over the loss rate of most practical networks. Additionally, with our choices of sender and receiver buffer sizes, we investigated all three interesting cases: (1) sender buffer size (Buf_S) < receiver buffer size (Buf_R), (2) $Buf_S = Buf_R$, and

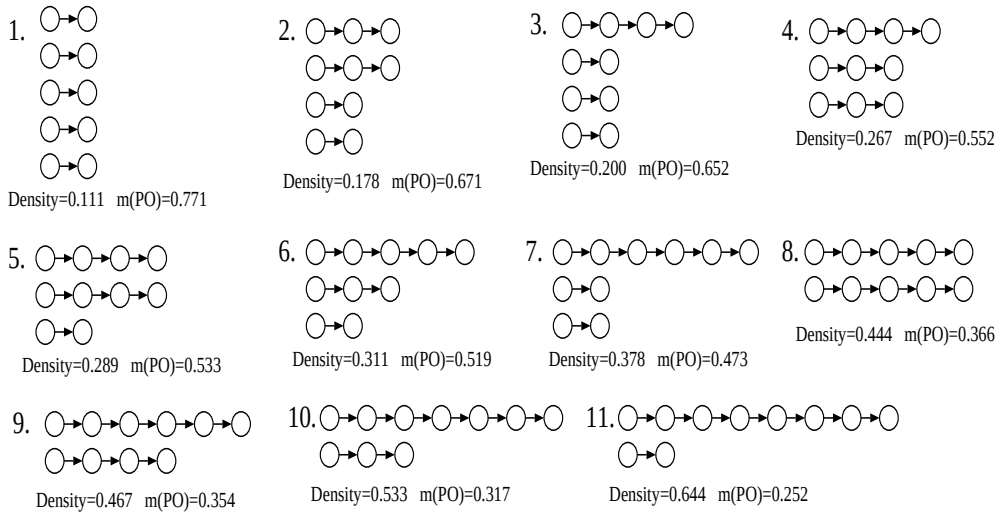

Figure 6: Parallel-Streams POs (Directed Graph Representation)

(3) $Buf_S > Buf_R$. When network layer delay is 4, a sender buffer size of 10 will be equal to the pipesize[3] (i.e., delay-bandwidth product of the system). Smaller buffer sizes represent the case when the pipesize is never full. By having two different delay values and three different buffer sizes, we studied a variety of cases in terms of network layer delays and Buf_S-to-pipesize ratios. We performed 36 simulation experiments, one for each combination of these system parameter values.

In each experiment, we simulated a total of 60 partial orders (40 POs in Figures 3-6 + 20 random POs). All POs were used in periodic form with 10 objects and 3,000 periods. A *periodic PO* can be defined as a partial order repeating itself some number of times. Periodic POs can be represented as $P^1 \oplus .. \oplus P^w$ where each P^i is identical, and w is the number of periods. The lossiness of the network layer in all experiments was modeled by a Bernoulli process. Our absolute results might differ for a bursty-loss process, however, we expect identical relative advantages in comparison of two partial order metrics. Additionally, in all experiments, constant object sizes were used. In general, given a PO with variable object sizes, we can obtain an equivalent PO with constant object size by breaking large objects into smaller ones that are chained to each other. Thus, using fixed object sizes for these experiments does not limit the effectiveness of our results.

The 36 experiments show that partially ordered services provide a throughput improvement only when the sender has many more buffers than the receiver. Since most transport layer protocols tend to use sender and receiver buffer sizes of roughly equal size, for most practical purposes, we can say that a partially ordered service provides no throughput improvement over an ordered service. Because of this, we focus on the correlation of partial order metrics to the other performance statistics (i.e., end-to-end packet delay, standard deviation of packet delay, and buffer utilization at receiver). Throughput correlation results are given only when larger sender buffer sizes are simulated

By studying the correlation of both metrics to expected performance in instances where all system parameters are fixed, we determine which of these metrics is useful for ordering PO's by expected performance (independent of network conditions).

[3]Notice that there is no advantage of using a larger sender buffer size than the pipesize since retransmissions get priority over new packets.

4.5 Simulation Results

	Dens	$m(PO)$	Dens	$m(PO)$	Dens	$m(PO)$	Dens	$m(PO)$	Dens	$m(PO)$	Dens	$m(PO)$
	loss rate = 0.01		loss rate = 0.05		loss rate = 0.1		loss rate = 0.2		loss rate = 0.4		loss rate = 0.6	
	Sender Buffer Size = 3											
$\overline{T_{end}}$	0.88	-0.89	0.91	-0.91	-0.92	-0.91	0.92	-0.92	0.93	-0.92	0.91	-0.91
$STD(T_{end})$	0.86	-0.88	0.90	-0.89	0.92	-0.90	0.92	-0.92	0.94	-0.89	0.90	-0.89
$\overline{R\_Buff}$	0.89	-0.90	0.91	-0.91	0.92	-0.92	0.92	-0.92	0.92	-0.92	0.91	-0.91
	Sender Buffer Size = 5											
$\overline{T_{end}}$	0.92	-0.88	0.95	-0.91	0.96	-0.93	0.96	-0.93	0.97	-0.94	0.95	-0.93
$STD(T_{end})$	0.92	-0.87	0.95	-0.90	0.96	-0.92	0.96	-0.91	0.96	-0.90	0.96	-0.88
$\overline{R\_Buff}$	0.93	-0.91	0.95	-0.93	0.96	-0.95	0.96	-0.95	0.96	-0.96	0.95	-0.94
	Sender Buffer Size = 10											
$\overline{T_{end}}$	0.92	-0.89	0.96	-0.92	0.97	-0.92	0.97	-0.92	0.97	-0.92	0.97	-0.93
$STD(T_{end})$	0.93	-0.90	0.97	-0.92	0.97	-0.90	0.95	-0.86	0.95	-0.87	0.96	-0.89
$\overline{R\_Buff}$	0.84	-0.84	0.87	-0.86	0.90	-0.88	0.95	-0.93	0.97	-0.96	0.97	-0.96
λ	-0.93	0.89	-0.96	0.91	-0.97	0.92	-0.97	0.92	-0.98	0.91	-0.97	0.90

Table 2: Correlation coefficients at selected fixed system conditions when network layer delay= 4

	Dens	$m(PO)$	Dens	$m(PO)$	Dens	$m(PO)$	Dens	$m(PO)$	Dens	$m(PO)$	Dens	$m(PO)$
	loss rate = 0.01		loss rate = 0.05		loss rate = 0.1		loss rate = 0.2		loss rate = 0.4		loss rate = 0.6	
	Sender Buffer Size = 3											
$\overline{T_{end}}$	0.89	-0.89	0.90	-0.90	0.92	-0.91	0.92	-0.91	0.92	-0.92	0.91	-0.91
$STD(T_{end})$	0.87	-0.88	0.89	-0.89	0.92	-0.90	0.93	-0.90	0.92	-0.91	0.91	-0.89
$\overline{R\_Buff}$	0.89	-0.89	0.91	-0.91	0.92	-0.92	0.92	-0.92	0.92	-0.92	0.91	-0.91
	Sender Buffer Size = 5											
$\overline{T_{end}}$	0.93	-0.89	0.95	-0.91	0.96	-0.93	0.97	-0.94	0.97	-0.94	0.95	-0.93
$STD(T_{end})$	0.93	-0.89	0.95	-0.90	0.97	-0.92	0.97	-0.91	0.97	-0.91	0.95	-0.91
$\overline{R\_Buff}$	0.94	-0.92	0.95	-0.94	0.96	-0.95	0.96	-0.96	0.96	-0.96	0.95	-0.95
	Sender Buffer Size = 10											
$\overline{T_{end}}$	0.92	-0.89	0.96	-0.93	0.97	-0.93	0.97	-0.92	0.97	-0.92	0.98	-0.93
$STD(T_{end})$	0.93	-0.91	0.97	-0.92	0.97	-0.88	0.95	-0.86	0.95	-0.88	0.97	-0.89
$\overline{R\_Buff}$	0.77	-0.78	0.80	-0.82	0.87	-0.86	0.94	-0.92	0.97	-0.96	0.97	-0.96
λ	-0.93	0.90	-0.96	0.92	-0.97	0.92	-0.97	0.92	-0.98	0.91	-0.97	0.90

Table 3: Correlation coefficients at selected fixed system conditions when network layer delay= 6

Table 2 introduces the correlation coefficients at various fixed loss rates and fixed buffer sizes when network layer delay= 4. Similarly, Table 3 presents the corresponding values for network layer delay= 6. These tables' correlation values indicate how good the partial order metrics are in correlating to the performance statistics for the 60 partial orders at given fixed system conditions. For example, the table entry 0.88 for density (noted as "**Dens**" in the table) and $\overline{T_{end}}$ indicates that ρ(density, $\overline{T_{end}}$) = 0.88 for the 60 POs when end-to-end packet delays obtained at Buf_S = 3, loss rate= 0.01, and network layer delay= 4 are considered.

It can be seen that both $m(PO)$ and density correlate highly with performance for a given set of network conditions. The observed correlation coefficients are much higher than even we expected. Both metrics would be ideal candidates for ordering partially ordered services along the x-axis of

Figure 1. Since density is significantly easier to compute for a given PO, it is the metric of choice for quantifying partially ordered services.

References

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

[2] Nate Baxter, Herman Chien, Andy Loreen, Kathryn Marshall, and Steven Baraniuk. *OPNET Manual*. MIL 3, Inc, 1993.

[3] G. Brightwell and P. Winkler. Counting Linear Extensions. *Order*, 8, 225–242, 1991.

[4] G. Brightwell and P. Winkler. Counting Linear Extensions is #P-Complete. In *Proceeedings of the 23rd ACM Symposium on the Theory of Computing*, 175–181, 1991.

[5] Thomas J. Connolly, Paul D. Amer, and Phillip T. Conrad. RFC-1693, An Extension to TCP: Partial Order Service.

[6] Phillip T. Conrad, Edward Golden, Paul. D. Amer, and Rahmi Marasli. A Multimedia Document Retrieval System Using Partially-Ordered/Partially-Reliable Transport Service. In *Multimedia Computing and Networking 1996 (MMCN96; sponsored by SPIE/IS&T)*, San Jose, CA, USA, Jan 1996.

[7] 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), 270–281, May 1994.

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

[9] William V. Gehrlein. On Methods for Generating Random Partial Orders. *Operations Research Letters*, 5(6), 285–291, December 1986.

[10] Rahmi Marasli, Paul D. Amer, and Phillip T. Conrad. Partial Order Transport Service: An Analytic Model. (Submitted for publication).

[11] Rahmi Marasli, Paul D. Amer, and Phillip T. Conrad. Retransmission-Based Partially Reliable Transport Service: An Analytic Model. In *IEEE INFOCOM'96*, San Fransisco, CA, March 1996. IEEE.

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

[13] Sheldon Ross. *A First Course in Probability, Fourth Edition*. Macmillian College Publishing, 1994.

[14] R. Stanley. *Enumerative Combinatorics: Volume 1*. Wardsworth + Brooks/Cole Advanced Books & Software, 1986.