Partially Reliable Transport Service *

Rahmi Marasli Paul D. Amer Phillip T. Conrad

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

Abstract

An analytic model is presented for a partially reliable *transport protocol based on retransmissions. The model illustrates tradeoffs between two QoS parameters (delay and throughput), and various levels of reliability. The model predicts that the use of reliable transport service when an application only needs a partially reliable one causes considerable throughput decreases and delay increases in lossy networks. On the other hand, over lossy networks, unreliable transport service is unable to respect an application's loss tolerance. In lossy environments, partially reliable transport service avoids the extra cost of reliable transport service, and, simultaneously, guarantees the minimal reliability that an application requires. Retransmission-based partially reliable transport service can be provided through either sender-based or receiver-based loss detection and recovery. Results show that both techniques provide almost identical reliability and delay. However, a sender-based approach provides better throughput than a receiver-based approach at high ack loss rates.*

1 Introduction

Many applications such as video and audio can tolerate loss. When the network layer only provides a best-effort service such as the Internet's IP protocol, the loss rate of the underlying network service may be higher than an application's tolerance for loss. In this case, the transport layer becomes responsible for enhancing the level of reliability provided to the application. This enhancement comes at the expense of other Quality of Service (QoS) parameters. For example, TCP enhances IP service to full reliability at the cost of increased delay and reduced throughput. UDP, on the other hand, introduces virtually no increase in delay or reduction in throughput, but provides no reliability enhancement over IP. To achieve the best tradeoff between reliability and other QoS parameters, partially reliable services have been proposed [1, 3, 4].

This paper analytically models *partially reliable* transport layer protocols. These protocols fill the gap between reliable and unreliable transport service by allowing an application to specify a *controlled* level of loss, and then enhancing an unreliable network service *just enough* to guarantee this loss level. Since partially reliable transport service does not insist on delivering all of the data, it can provide higher throughput and lower delay than reliable transport service, while, at the same time, respecting an application's loss tolerance.

Although widely rejected, the retransmission of continuous media (e.g., audio) is shown to be feasible by Dempsey [3]. Dempsey's results are encouraging in terms of providing partially reliable transport service through retransmissions for multimedia applications. Dempsey mainly uses increased control time[1] to allow timely retransmissions of the lost data.

Partial reliability guarantees can be provided through transport layer retransmissions. The transport protocol makes only enough retransmissions to satisfy the loss tolerance of the application. In providing partially reliable service, the transport layer must first *detect* a lost packet and then *decide* whether or not to *recover* from the loss. A lost packet can be detected by either the receiver or the sender. The recovery decision is then performed by consulting the application's loss tolerance so that the reliability QoS guarantee will be met. Once the transport layer detects a lost packet and decides to recover it, the lost packet is retransmitted. Depending on which transport entity (i.e., the sender or the receiver) detects and decides to recover, two basic techniques are possible:

- **Sender-based loss detection and recovery:** The sender is responsible for detecting and deciding to recover lost data. Lost data detection is done mainly through timers and occasionally with negative acks. Once a lost packet is detected, the sender decides whether or not to retransmit it based on the loss tolerance of the application.
- **Receiver-based loss detection and recovery:** The receiver detects lost data through gap-detection and loss-timers [4]. Once a lost packet is detected and the recovery decision is made, the receiver requests retransmission of the lost packet by returning a negative ack to the sender.

In previous work, the authors investigated the cost of not

*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.

[1] Control time is the time that the first packet in a continuous media (e.g., audio) stream is artificially delayed at the receiver in order to buffer sufficient packets to provide for continuous playback in the presence of jitter [3].

using the ideal reliability service for applications [6]. This investigation uses a sender-based method called *k_XMIT* reliability. In contrast, in this paper, we analytically study both sender-based and receiver-based methods in providing partially reliable service, and compare their performances. Additionally, this paper uses a technique other than *k_XMIT* reliability to characterize the recovery decision of the transport layer.[2] The analytic study serves the following purposes: (1) To show the performance gains of partially reliable transport service over reliable transport service for applications that can tolerate loss. These gains demonstrate the penalty that a current application pays by using reliable service (e.g., TCP) when a partially reliable service would suffice. (2) To compare the performance of sender-based and receiver-based methods in providing partially reliable services, and to determine the effects, if any, of the three factors studied (i.e., packet losses, ack losses, or applications' loss tolerance) on the performance of the two methods. This result is helpful in designing partially reliable transport protocols.

The paper is organized as follows: Section 2 introduces a model for a sender-based approach and shows the tradeoffs between reliability and other QoS parameters. In Section 3, a receiver-based approach is studied and the two basic approaches are compared. Related work is introduced in Section 4, and the main results are summarized in Section 5.

2 Sender-Based Loss Recovery

We present an analytic model of partially reliable transport service using sender-based loss detection and recovery. This model is similar to the one presented in [5] to study the protocol Partial Order Connection (POC).[3] This model differs in that it does not require *all* of the transmitted objects to eventually be communicated. In [5], only full reliability is considered.

Results confirm our expectation that for an application that can tolerate some loss, partially reliable transport service provides better throughput and delay than reliable transport service —in general, increasingly so as the underlying network service gets more lossy and as an application's loss tolerance increases.

2.1 Introduction to Model

To abstract partially reliable transport service's usage, we use a three layer architecture: the network layer, the transport layer, and the user application layer (see Figure 1.A).

The transport layer enhances the network's assumed unreliable service (i.e., uncontrolled loss) into a partially reliable service (i.e., controlled loss) by using an ARQ-like sender-based loss detection and recovery. It does so as follows: Transport 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 arrive within its timeout period, Transport Sender assumes the packet is lost. In the study of both sender-based and receiver-based methods, we model the transport protocol's decision on recovering a lost packet via recovery probabilities. When a lost packet is detected, Transport Sender decides to retransmit the packet with sender recovery probability (α_s). Modeling the recovery decision probabilistically facilitates comparison between the sender-based and receiver-based models. In practice, one could imagine that an application itself determines the importance of recovering a given lost packet, and communicates this information to the transport layer.

By assumption, User Sender submits constant size packets to Transport Sender. If necessary, large packets are assumed fragmented into small ones at User Sender, and the fragmented packets are assembled back to the original ones at User Receiver.

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 there is no problem with running out of buffer space at Transport Receiver.[4] These simplifying assumptions, while in some cases strong, are needed for the mathematical analysis of the model. The results obtained under these assumptions are useful in *comparing* various types of service, and in analyzing the *trends*, since we expect the effects of these assumptions to be similar across various levels of reliability (i.e., reliable, partially reliable, unreliable), and for both sender-based and receiver-based methods.

The following notation is used throughout the paper: t_{pack} represents packet transmission time. t_{out} represents the timeout period and is equal to round-trip delay. t_{delay} represents the one-way propagation delay. p and q are packet and ack loss probabilities within Unreliable NET, respectively. p_{succ} represents the probability of a successful packet transmission and ack transmission, i.e., $p_{succ} = (1-p)*(1-q)$. Finally, Buf_S and Buf_R are number of buffers at Transport Sender and Transport Receiver, respectively.

2.2 Definitions of Target Values

The throughput and the delay characteristics of partially reliable transport service will be analyzed by computing the set of target values defined in Table 2.

Figure 1.B shows a more detailed model of our system. λ_{US} and λ_{UR} represent the admission rate at the sender and the throughput at the receiver, respectively. For reliable (i.e., no loss) service, we have: $\lambda_{US} = \lambda_{UR} = \lambda_{Reliable}$. In [5], it is shown that $\lambda_{Reliable} = \frac{p_{succ}}{t_{pack}}$. In further sections, we use this result when comparing partially reliable service to reliable service.

The transport layer delay, *Delay*, is the expected time for a packet to arrive at Transport Receiver once given to Transport Sender. This delay does not include any buffering time at Transport Receiver. *Delay* will only be computed for packets that are successfully received by Transport Receiver

[2] See Section 4 for a more detailed comparison of the two studies.

[3] POC is a proposed transport-layer protocol that provides *partially ordered* and *partially reliable* service to its users. POC fills the gap between *ordered and reliable* (e.g., TCP) and *unordered and unreliable* (e.g., UDP) services [1, 2].

[4] See Table 1 for the full set of 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	$Buf_S = \frac{t_{out}}{t_{pack}}$ and $Buf_R = \infty$
7	Only selective acks are used (either positive or negative)
8	All packets are ready at User Sender waiting to be communicated, or equivalently, a packet arrives at User Sender at every t_{pack} time
9	Transport Sender retransmits detected lost packets with probability α_s

Table 1: Assumptions

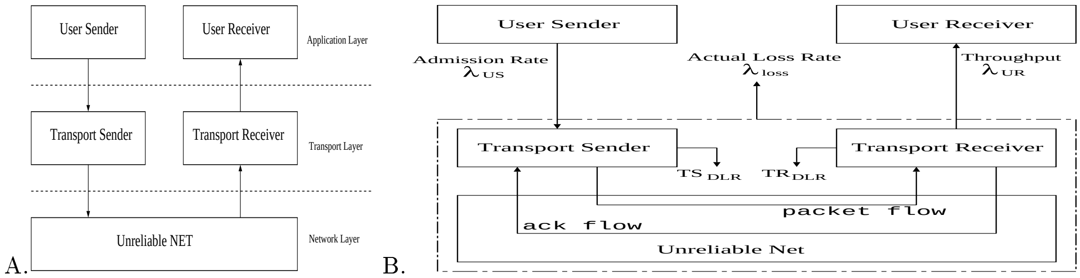


Figure 1: (A) Architecture; (B) Packet Flow Rate Among Different Layers

since User Receiver (i.e., the receiving application) cannot quantify the delays for lost packets.

In this paper, we focus on the investigation of delivery probability (PLD), throughput (λ_{UR}), and transport layer delay ($Delay$). Using these target values, we compare partially reliable services with reliable and unreliable services. The other target values listed in Table 2 are included for the sake of completeness.

2.3 Computation of Target Values for Sender-Based Method

The analysis of a sender-based approach will be done by computing delivery probability (PLD), throughput (λ_{UR}), and transport layer delay ($Delay$).

For a sender-based method, the possible states in relation to the condition of a packet together with the transition probabilities are given in Figure 2.A.[5] If the packet transmission succeeds (with probability $1-p$), then the diagram enters **Success**, representing the successful reception of the packet by Transport Receiver. If the packet transmission fails with probability p, or the corresponding ack is lost in the network layer with probability q, then the diagram enters **Loss Detection**, meaning that Transport Sender detects the lost packet. A lost packet is retransmitted with probability α_s. If Transport Sender receives an ack with probability $1-q$ or decides not to retransmit a lost packet with probability $1-\alpha_s$, then the diagram enters **End**, representing the release of the packet from the sender's buffers.

[5]Note that this is not a state diagram of either Transport Sender or Transport Receiver, but is a diagram representing the status of each packet in the system.

2.3.1 Delivery Probability: PLD

Delivery probability, PLD, is the probability that a packet from User Sender is delivered to its destination by the transport layer. PLD can also be seen as the probabilistic delivery guarantee provided by partially reliable service. Thus, the reliability guarantee of the transport layer is determined by this probability. We can compute PLD as the probability of going from **Transmission** to **Success**.[6]

$$PLD = \frac{1-p}{1-(p*\alpha_s)} \tag{1}$$

It is noteworthy that PLD is independent of the ack loss rate (i.e., q). Whether *none* or *all* of the acks are lost, the delivery probability does not change. Expression (1) shows that a sender-based method can provide reliability guarantees regardless of the ack loss level. Intuitively, this is because Transport Sender can detect a lost packet through its timers without relying on the responses from Transport Receiver.

As expected, $PLD = 1-p$ for unreliable service (i.e., $\alpha_s = 0$) and $PLD = 1$ for reliable service (i.e., $\alpha_s = 1$). Thus, delivery probability cannot be smaller than the packet success rate (i.e., $1-p$). Additionally, if the packet loss rate of the underlying network layer is greater than an application's tolerance for loss, then unreliable transport service cannot respect the loss tolerance of the application.

2.3.2 Throughput: λ_{UR}

Throughput, λ_{UR}, is the rate at which the receiving application (i.e., User Receiver) gets packets. Some applications may need certain throughput QoS guarantees.

[6]Computational details can be found in [7].

Target Value	Definition
Delivery Probability (PLD)	P(delivering a packet to User Receiver)
Throughput of Reliable Service ($\lambda_{Reliable}$)	Average number of packets that are delivered per unit time at Transport Receiver when nothing is permitted to be lost
Transport Sender Declared Loss Rate (TS_{DLR})	Average number of packets that are detected to be lost but not retransmitted by Transport Sender per unit time
Transport Receiver Declared Loss Rate (TR_{DLR})	Average number of packets that are detected to be lost but not requested to be retransmitted by Transport Receiver per unit time
Throughput (λ_{UR})	Average number of packets that are delivered per unit time from Transport Receiver to User Receiver
Admission Rate (λ_{US})	Average number of packets that are given from User Sender to Transport Sender per unit time
Actual Loss Rate (λ_{loss})	Average number of packets that are given to Transport Sender for transmission but not delivered to User Receiver per unit time, defined as $\lambda_{US} - \lambda_{UR}$
Transport Layer Delay ($Delay$)	Expected transport layer delay defined as the time from a packet's first transmission to the time it is successfully received by Transport Sender

Table 2: Target Values

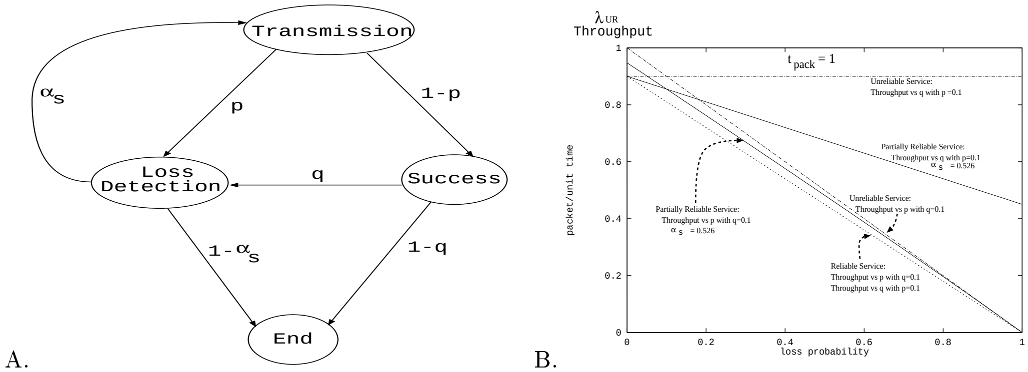


Figure 2: (A) Sender-Based Loss Detection and Recovery; (B) Effects of loss probabilities on λ_{UR}

Let *TS_Time* be the time that a packet spends at Transport Sender. *TS_Time* can be computed as the expected time to go from **Transmission** to **End** since **End** depicts the release of the packet from the sender's buffers. Then, $TS\_Time = \frac{t_{out}}{1-(1-p_{succ})*\alpha_s}$. With assumptions 2, 3, 4, 5, 6, and 8, the number of packets at Transport Sender is always Buf_S. Thus, by using Little's theorem and *TS_Time*, the admission rate, λ_{US}, is: $\lambda_{US} = \frac{1-(1-p_{succ})*\alpha_s}{t_{pack}}$. λ_{UR} can be obtained by the product of λ_{US} and PLD:

$$\lambda_{UR} = \frac{1-(1-p_{succ})*\alpha_s}{t_{pack}} * \frac{1-p}{1-(p*\alpha_s)} \qquad (2)$$

Expression (2) shows that partially reliable transport service provides better throughput than reliable transport service (i.e., $\lambda_{UR} > \lambda_{Reliable}$) whenever there are ack losses in the network layer (i.e., $q \neq 0$). Intuitively, this can be explained as follows. Consider the case of a packet successfully arriving at Transport Receiver. That packet will be delivered to User Receiver and an ack for it will be returned to Transport Sender. If the ack is lost in the network layer, the corresponding delivered packet always will be retransmitted in a reliable service. On the other hand, such unnecessary retransmissions will be avoided with probability $1-\alpha_s$ in a partially reliable service. For reliable service, $\lambda_{UR} = \frac{p_{succ}}{t_{pack}}$, and for unreliable service, $\lambda_{UR} = \frac{1-p}{t_{pack}}$. Therefore, the maximal throughput improvement by any partially reliable service over reliable service is bounded by $\frac{(1-p)*q}{t_{pack}}$.

The relationship between λ_{UR} and loss probabilities (either p or q depending on the graphed curve) is investigated in Figure 2.B. This figure illustrates "λ_{UR} vs p and q". Packet and ack losses have different effects on the throughput of partially reliable services. In general, λ_{UR} decreases as either p or q increases, but the decrease in λ_{UR} is slower with increasing q than with increasing p. *Thus, in a partially reliable service, ack losses are less detrimental to throughput than packet losses.*

Figure 2.B also shows that increasing reliability from unreliable to partially reliable to reliable, when using a network where acks are lost, can result in severe throughput drops. If there are low ack losses (e.g., $q \leq 0.1$), then the degradation in throughput as packet losses increase is generally independent of the system reliability, that is, whether or not the system provides reliable, partially reliable or unreliable service. However, if there are high ack losses (e.g., $q \geq 0.2$), then the degradation in throughput is significantly greater for reliable service than for partially reliable service and in turn than for unreliable service. *In general, the more a network loses acks when an application can tolerate loss, the more costly in terms of throughput it is to use reliable service instead of partially reliable service.*

2.3.3 Transport Layer Delay: *Delay*

Transport layer delay, *Delay*, is the expected time for a packet to arrive at Transport Receiver once it is given to Transport Sender by User Sender. Applications may need certain delay QoS guarantees. For many applications such as real time audio and video, lower delay is more important than higher throughput.

It is important to note that *Delay* is only computed for those packets that are successfully received by Transport Receiver. The delay of packets that are never received and are dropped are not included in overall *Delay* computation. By using Figure 2.A, *Delay* can be computed as the expected time to go from **Transmission** to **Success**, which is:

$$Delay = t_{delay} + \frac{p * \alpha_s}{1 - (p * \alpha_s)} * t_{out} \qquad (3)$$

As expected, $Delay = t_{delay}$ for unreliable service, and $Delay = t_{delay} + \frac{p}{1-p} * t_{out}$ for reliable service. Expression (3) shows that the *Delay* of partially reliable service increases and converges to that of reliable service with increasing α_s. For reliable and partially reliable service, *Delay* increases as packet losses increase. However, the delay increase is slower for partially reliable service than for reliable service. *In general, the more a network loses packets, the greater will be the penalty in increased delay by using reliable service rather than a partially reliable one.*

3 Receiver-Based Loss Recovery

In this section, we present an analytic model for providing partially reliable transport service using receiver-based loss detection and recovery. We also compare the performance of receiver-based and sender-based methods. We determine the effects, if any, of three factors studied (i.e., packet loss, ack loss, or application's loss tolerance) on the performance of two basic methods in providing partially reliable service.

3.1 Introduction to Model

In a receiver-based model, Transport Receiver, not Transport Sender, detects and decides to recover any packets lost by the network layer. The user and network layers of this model are identical to those of the sender-based model.

The transport layer provides partially reliable service as follows: Transport Sender takes a packet from User Sender, transmits the packet over the network, buffers the packet, and waits for a response from Transport Receiver. For each successfully received packet, Transport Receiver sends a selective positive ack (PACK) to Transport Sender. When Transport Receiver detects a lost packet (through either gap detection or loss timers), it decides to recover the lost packet with recovery probability (α_r). If a recovery is desired, Transport Receiver requests retransmission of the lost packet by sending a selective negative ack (NACK) to Transport Sender. With probability $1 - \alpha_r$, Transport Receiver decides not to recover a lost packet and sends a PACK for it to Transport Sender. When Transport Sender receives a PACK, it releases the corresponding packet from its buffers.

In practice, Transport Receiver initially detects a lost packet via gap detection. Gap detection for packet i occurs when Transport Receiver correctly receives a packet j such that j was transmitted after i (i.e., $j > i$). In our model, however, we assume Transport Receiver detects lost packet i even sooner at time "$t_{delay} = t_{pack} +$ one way network delay" after packet i's transmission has started. It would actually take longer than t_{delay} for a packet to be detected lost at Transport Receiver. As a result of this assumption, a lost packet is retransmitted t_{out} after its original transmission whenever Transport Receiver decides to recover it and the NACK is not lost.

Furthermore, after sending a NACK, Transport Receiver sets a timer. If the corresponding packet does not arrive within t_{out}, Transport Receiver behaves as if the packet was lost again.

Since Transport Sender frees its buffers only when a PACK is received, PACKs should be sent reliably. To achieve reliable PACK transmission, (1) for each correctly received PACK, Transport Sender sends a special packet (called PACK-INFORM) to Transport Receiver, and (2) Transport Receiver continues to send a PACK at every t_{out} until a corresponding PACK-INFORM is received. Since PACK-INFORMs are small packets, they can be piggybacked in a packet flowing to Transport Receiver.

In the analysis, it is assumed that PACK-INFORMs always arrive safely to Transport Receiver regardless of the network conditions. Because of this assumption, our computations will bias in favor of a receiver-based method especially in throughput comparisons; yet, as we shall see, a sender-based method still provides better throughput at high ack loss rates and small sender buffer sizes.

Real implementations of receiver-based methods (e.g., [4]) generally use cumulative, not selective, PACKs to inform Transport Sender that it is unnecessary to keep the corresponding packet in its buffers any more. With cumulative PACKs, Transport Receiver may not need to resend a PACK when it is lost, because with the next cumulative PACK to be transmitted, the lost information in the previous PACK(s) will be included. As a result, with cumulative PACKs, it may be unnecessary to have PACK-INFORMs in the system. For these reasons, we study the effects of PACK and NACK losses on the system performance separately.

Computational results show that the assumptions related with PACKs and PACK-INFORMs only affect λ_{UR}, not *PLD* and *Delay*. Thus, whether PACKs are selective or cumulative, and whether or not PACK-INFORMs are used, our results related with *PLD* and *Delay* will remain unchanged. While comparing the throughputs of sender-based and receiver-based methods in Section 3.2.2, we choose loss levels for the packets and acks such that the values will be fair to both methods.

The following assumptions are different in this model from the assumptions in Section 2: (1) Transport Receiver can make a lost packet detection at time t_{delay} after a packet is sent if it is lost, or at time t_{out} after a NACK is sent if the NACK is lost. (2) The probability of losing a PACK and a NACK are r and s, respectively. These probabilities are constant and independent for each ack transmission.[7] (3) The probability of losing a PACK-INFORM in the network layer is zero.

3.2 Computation of Target Values for Receiver-Based Method

The analysis of a receiver-based approach is similar to that of sender-based one (see Section 2.3). We derive formulae for delivery probability (PLD), throughput (λ_{UR}) and transport layer delay ($Delay$), and compare these values with the ones derived for a sender-based method.

For a receiver-based method, the possible states related with the condition of a packet together with the transition probabilities are given in Figure 3.A (see Figure 2.A for the corresponding sender-based diagram). **Transmission** represents the packet transmission. If the packet is lost with probability p, or corresponding NACK is lost with probability s, then the diagram enters **Loss Detection**, meaning that Transport Receiver detects the lost packet. If the packet transmission succeeds with probability $1-p$, or Transport Receiver decides not to recover the lost packet with probability $1-\alpha_r$, then the diagram enters **PACK Send**. On the other hand, if Transport Receiver decides that it is still necessary to recover the lost packet with probability α_r, then the diagram enters **NACK Send**. If NACK transmission succeeds with probability $1-s$, then the corresponding packet will be retransmitted. The diagram departs from **PACK Send** state only with a successful PACK transmission (with probability $1-r$). Transport Sender releases a packet from its buffers in **End** when a PACK for the packet is received.

Notice that PLD is the probability of going from **Transmission** to **PACK Send** without passing through **Loss Detection** $\rightarrow$ **PACK Send** transition. Furthermore, TS_Time can be computed by finding the expected time to go from **Transmission** to **End**. Since the number of packets at Transport Sender is always Buf_S, λ_{US} can easily be computed by Little's theorem. λ_{UR} is simply the product of λ_{US} and PLD. $Delay$ is the expected time to go from **Transmission** to **PACK Send** without passing through **Loss Detection** $\rightarrow$ **PACK Send** transition. Based on these observations, the computations of the target values are given in Expressions (4) through (8).

$$PLD = \frac{(1-p)*(1-s*\alpha_r)}{1-(p+s-p*s)*\alpha_r} \quad (4)$$

$$TS\_Time = \left(\frac{1}{1-r} + \frac{p*\alpha_r}{1-(p+s-p*s)*\alpha_r}\right)*t_{out} \quad (5)$$

$$\lambda_{US} = \frac{(1-r)*(1-(p+s-p*s)*\alpha_r)}{1-\alpha_r(s+p*r-p*s)} * \frac{1}{t_{pack}} \quad (6)$$

$$\lambda_{UR} = \frac{(1-r)*(1-p)*(1-s*\alpha_r)}{1-\alpha_r(s+p*r-p*s)} * \frac{1}{t_{pack}} \quad (7)$$

$$Delay = t_{delay} + \frac{p*\alpha_r}{1-(p+s-p*s)*\alpha_r} * t_{out} \quad (8)$$

Expressions (4) and (8) show that PACK loss probability (i.e., r) does not affect PLD nor $Delay$. Intuitively, this can be explained by the fact that the target values PLD and $Delay$ are only concerned with the safe arrival of a packet at Transport Receiver. The correct arrival of PACK at the sender, the release of the corresponding packet from Transport Sender's buffers, and the safe arrival of the corresponding PACK-INFORM to Transport Receiver have no impact on these target values.[8] Thus, the results related with PLD and $Delay$ are unaffected by any assumption related with PACKs and PACK-INFORMs.

Expressions (1)-(3) of sender-based method and expressions (4)-(8) of receiver-based method show that for zero ack loss rate (i.e., $r = s = q = 0$), sender-based and receiver-based methods perform identically. That is, with no ack losses in the system, both methods achieve the same reliability, throughput, and delay at any packet loss rate and any application's loss tolerance.

3.2.1 Delivery Probability: PLD

Unlike the delivery guarantee of a sender-based method, PLD in a receiver-based method depends not only on the packet loss rate but also the NACK loss rate. This is because the retransmission of a lost packet in a receiver-based method is initiated by a successful NACK transmission.

Figure 3.B gives an example relationship between PLD and recovery probabilities. This figure shows that sender-based and receiver-based methods provide virtually identical delivery probabilities for any $\alpha_s = \alpha_r$ values when the loss level is 0.1. Figure 4.A, illustrating "PLD vs p", shows that the decrease in PLD with increasing packet losses is almost identical for the two methods. Similarly, as the "PLD vs s" graph of Figure 4.B illustrates, a sender-based method provides only slightly better delivery probability than a receiver-based one, particularly as NACK losses increase. Such differences are negligible at practical loss levels (i.e., loss level ≤ 0.1). For example, for $p = s = 0.1$ and $\alpha_s = \alpha_r = 0.5$, the PLD difference between the two methods is less than 0.3%. *Thus, in general, both methods provide almost identical reliabilities when they are given the same chance of recovering lost packets (i.e., $\alpha_s = \alpha_r$).*

3.2.2 Throughput: λ_{UR}

The effects of PACK and NACK losses on a receiver-based method's throughput are studied in Figure 5.A. This figure shows that PACK losses can be detrimental to throughput. In the figure, it is also clear that the impact of NACK losses on throughput can be ignored when compared to that of PACK losses. Thus, the effect of NACK losses on λ_{UR} is negligible while that of PACK losses is important.

Throughputs of sender and receiver-based methods are compared in Figures 6.A and 6.B. In these figures, all ack loss probabilities are taken to be equal (i.e., $s = r = q$). An ack in a sender-based method is equivalent to a PACK in a receiver-based method. They both inform sender that a corresponding packet is no longer needed at the receiver and that packet can be released from the sender's buffers. Thus,

[7] The primary reason for having different probabilities for PACK and NACK losses (i.e., r and s) is to study the effects of PACK and NACK losses separately on system performance. In practice, one might expect the network layer to lose PACKs and NACKs with equal probability.

[8] One also can easily see this result through Figure 3.A. The transitions that originate from **PACK Send** do not have any impact on PLD and $Delay$.

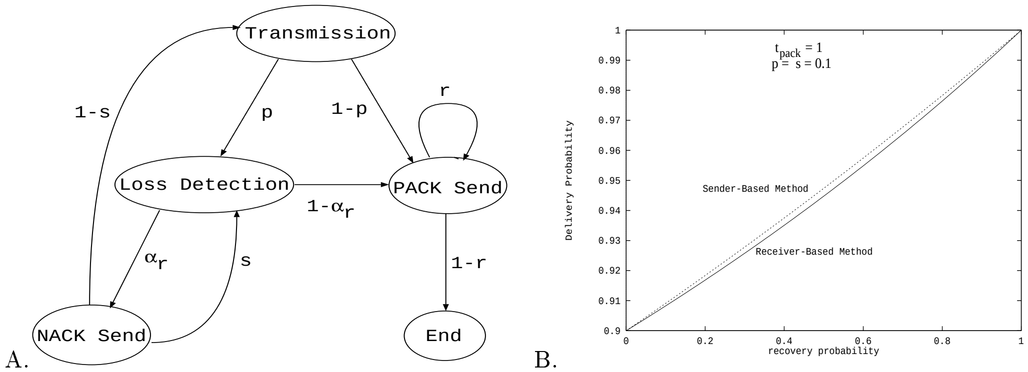


Figure 3: (A) Receiver-Based Loss Detection and Recovery; (B) PLD vs recovery probability

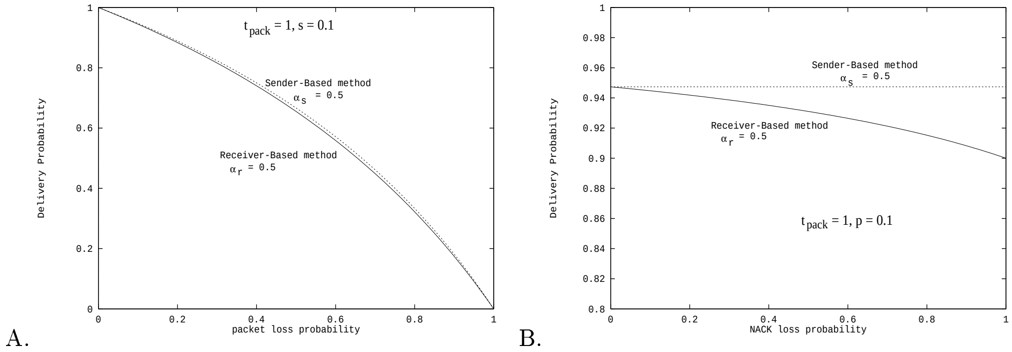


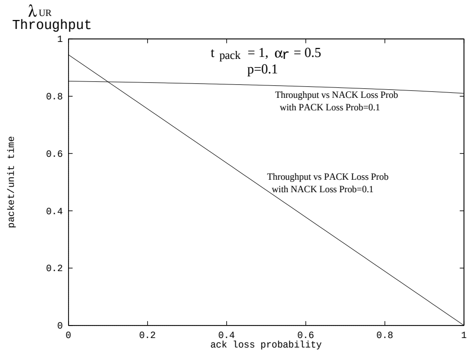
Figure 4: PLD vs loss probabilities

by letting $r = q$, we have a fair comparison between these two systems. Since the effects of NACK losses are negligible when compared to that of PACK losses, having $s = r$ will not affect our results. Hence, when $s = r = q$, we have a fair comparison between the throughputs of sender-based and receiver-based methods.

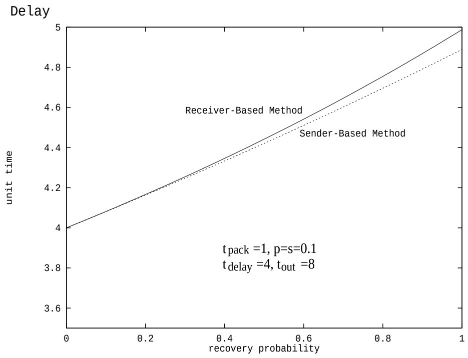
Figure 6.A illustrates "throughput vs recovery probabilities." This figure shows that a sender-based method provides higher throughput than a receiver-based one —increasingly so as the recovery probabilities decrease (or, equivalently, as the loss tolerance of the application increases). Figure 6.B studies the relationship between throughput and loss probabilities (i.e., p, r, s and q). The negative effects of packet losses are almost identical on the throughputs of both methods. On the other hand, ack losses have different impact. As ack losses increase, the throughput of sender-based method decreases slower than that of receiver-based method. Intuitively, we can explain this result as follows: in a receiver-based method, Transport Sender frees its buffers only when a PACK is received. Thus, if a PACK is lost, a buffer space at the sender will be occupied longer by a packet that is no longer needed by the receiving application. In a sender-based method, however, such a packet will be released from the buffers with probability $1 - \alpha_s$. Thus, in a sender-based method, more buffer space at Transport Sender will be made available for packets waiting to be transmitted by User Sender. Hence, a sender-based method will have higher admission rate and throughput than a receiver-based one at higher ack loss rates. For example, for $p = s = r = q = 0.1$ and $\alpha_s = \alpha_r = 0.5$, the throughput of sender-based approach is about 5.8% higher than that of receiver-based one.

By assumption, in the computations, the buffer size at the sender ($Buf_S = \frac{t_{out}}{t_{pack}}$) is equal to the pipeline size (i.e., the bandwidth-delay product). In a sender-based approach, we cannot take advantage of any larger buffer size at Transport Sender because a packet either receives its ack or timeouts after one roundtrip-delay. On the other hand, a receiver-based method can improve its throughput by increasing the sender buffer size beyond the pipeline size.[9] Based on these observations, we can say that *a sender-based method provides somewhat higher throughput than a receiver-based one in the presence of ack losses and for situations where the buffer size at the sender is smaller than the round trip delay-bandwidth product of the system.* Conversely, a receiver-based approach achieves higher throughput with larger sender buffer sizes and smaller ack losses.

[9] For example, in [4], it is shown that for a receiver-based approach and 100% delivery guarantee (i.e., reliable service), the maximal throughput can be achieved by a buffer size equal to the pipeline size at 0.01% packet loss level, whereas the same throughput can be achieved by a buffer size equal to approximately 2.5 times the pipeline size at 1% packet loss level. These results are obtained under the assumptions of no ack losses, no loss of retransmitted packets, and using flow control.

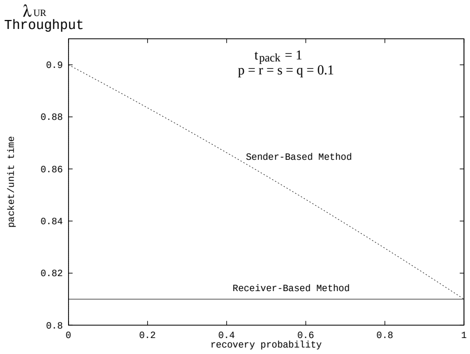


A.

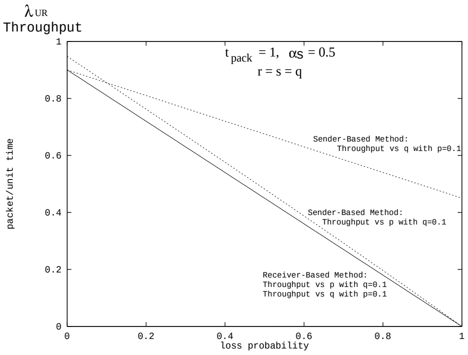


B.

Figure 5: (A) λ_{UR} vs ack loss probabilities; (B) $Delay$ vs recovery probabilities



A.



B.

Figure 6: Recovery and loss probabilities vs λ_{UR} (Throughput)

3.2.3 Transport Layer Delay: *Delay*

$Delay$ in a receiver-based approach depends on both packet and NACK losses. $Delay = t_{delay}$ for unreliable service and $Delay = t_{delay} + \frac{p}{(1-p)*(1-s)} * t_{out}$ for reliable service.

Figure 5.B illustrates "$Delay$ vs recovery probabilities" for the situation where timeout period (t_{out}) is twice the one-way delay (t_{delay}). Notice that for most practical cases, t_{out} will be roughly equal to $2*t_{delay}$. This figure shows that when loss level= 0.1, there is a negligible difference between the $Delay$ of the two methods at all values of $\alpha_s = \alpha_r$. Thus, regardless of the recovery probabilities (and hence the application's loss tolerance), sender-based and receiver-based methods provide virtually identical delay values at 10% network loss rate.

Figures 7.A and 7.B plot "$Delay$ vs p" and "$Delay$ vs s", respectively. These graphs illustrate the relative impacts of losing packets or losing NACKs on delay, respectively. The increase in $Delay$ as packet loss rate increases is essentially identical for both methods. On the other hand, with increasing NACK losses, a receiver-based method provides slightly worse (i.e., larger) delay than a sender-based one. Such differences, however, are negligible at practical loss levels (i.e., loss level ≤ 0.1). For example, for $p = s = 0.1$, $t_{out} = 2 * t_{delay}$ and $\alpha_s = \alpha_r = 0.5$, the $Delay$ difference between the two approaches is less than 0.48%.

In general, Section 3.2 shows that out of the three factors studied (i.e., packet loss, ack loss and an application's loss tolerance), only ack losses make a difference in the performance of the two methods, and mainly in the throughput comparisons.

4 Related Work

In the literature, Application-Oriented Error Control (AOEC) [4] and Partially Error-Controlled Connections (PECC) [3] provide partially reliable service by a receiver-based approach. AOEC has the objective of satisfying an application's loss tolerance with minimum retransmission overhead. AOEC guarantees that maximum loss tolerance of the application is always respected by retransmitting lost data whenever necessary. The analytic study of AOEC [4] assumes that no ack losses, no loss of retransmitted packets, and a 100% delivery guarantee (i.e., not a partially reliable service).

PECC is introduced to enable limited recovery of packet losses for stream-based communications in which data completeness must be traded off for low delay service. Unlike AOEC where emphasis is on loss tolerance, PECC gives priority to the delay constraints of the packets. The basic idea of PECC is applied to audio streams by Dempsey [3]. The new delay-constrained receiver-based loss detection and recovery scheme is called *Slack ARQ*. Dempsey shows that contrary

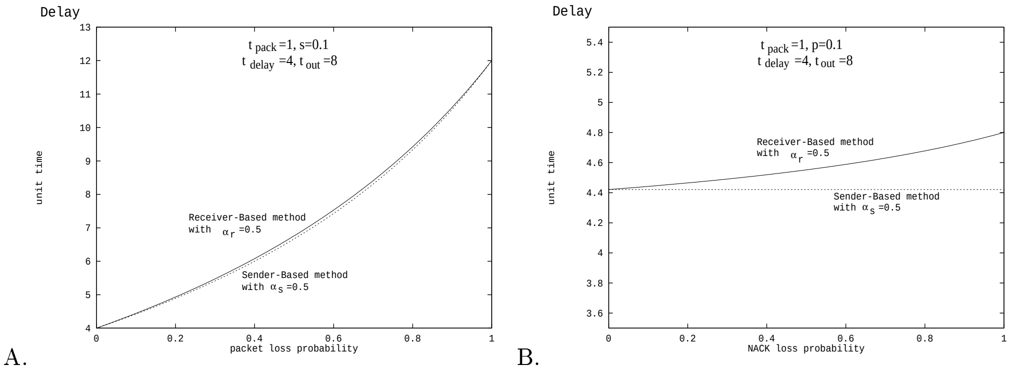


Figure 7: *Delay* vs loss probabilities

to the conventional wisdom, it is feasible to retransmit audio packets. *Slack ARQ* uses larger control time at the receiver to an extent that creates extra time for retransmissions once a packet is detected to be lost. Dempsey's results are encouraging in terms of providing partially reliable transport services through retransmissions for multimedia applications.

Neither of the existing analytic models (i.e., [3] or [4]) studies the effects of ack losses on the performance of partially reliable services. In this paper, we model the effects of both packet and ack losses as well as various levels of application's loss tolerance. Our results show that ack losses are more damaging to the throughput of a receiver-based approach than that of a sender-based one.

Reference [6] analyzes the cost of not using ideal reliability service for applications. This reference uses a sender-based method called *k_XMIT* reliability. A packet with *k_XMIT* reliability can be transmitted (original plus retransmissions) at most k times. After the k^{th} transmission, the packet will be released from Transport Sender's buffers without waiting for its ack. Three cost functions associated with the level of reliability that a system can support are introduced. The cost functions represent the penalty paid when the underlying transport service does not support the ideal reliability level for an application. The three cost functions are (1) throughput and (2) delay costs of using more reliability, and (3) loss cost of using less reliability. In contrast, this paper uses *recovery probabilities* to model sender-based and receiver-based methods, and to compare their performances. Using recovery probabilities is a better way of modeling than using *k_XMIT* because with recovery probabilities, one can characterize any application's loss tolerance, whereas with *k_XMIT* reliability, only discrete loss tolerance points can be represented. Also, modeling through recovery probabilities facilitates comparison of sender-based and receiver-based methods. On the other hand, the delay constraints of the packets can be more easily represented by *k_XMIT* reliability than by recovery probabilities.

5 Summary

This paper studies retransmission-based partially reliable transport service. Models illustrate the tradeoffs that are possible between two QoS parameters (delay and throughput), and various levels of reliability for both a sender-based and a receiver-based approach.

The models predict that the use of reliable transport service when an application only needs a partially reliable transport service causes considerable throughput decreases and delay increases in lossy networks. On the other hand, at high loss rates, unreliable transport service is unable to respect an application's loss tolerance. Thus, in lossy environments, partially reliable transport service is useful to avoid the extra cost of reliable transport service, and, at the same time, to guarantee the minimal reliability that an application requires.

The comparative study of sender-based and receiver-based methods shows that both methods provide almost the same reliability and delay. On the other hand, a sender-based method provides better throughput than a receiver-based one at high ack loss rates. Thus, out of the three factors studied (i.e., packet loss, ack loss and an application's loss tolerance), only ack losses make a difference in the performance of the two methods, and mainly in the throughput comparisons.

References

[1] P. Amer, C. Chassot, T. Connolly, P. 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] T. J. Connolly, P. D. Amer, and P. T. Conrad. RFC-1693, An Extension to TCP: Partial Order Service.

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

[4] F. Gong and G. Parulkar. An Application-Oriented Error Control Scheme for High-Speed Networks. Tech Report WUCS-92-37, Department of Computer Science, Washington University in St. Louis, November 1992.

[5] R. Marasli, P. D. Amer, and P. T. Conrad. An Analytic Study of Partially Ordered Transport Services. *Computer Networks and ISDN Systems.* (To appear).

[6] R. Marasli, P. D. Amer, and P. T. Conrad. Retransmission-Based Partially Reliable Transport Service: An Analytic Model. In *IEEE INFOCOM'96*, 621–629, San Fransisco, CA, March 1996.

[7] R. Marasli. *Partially Ordered and Partially Reliable Transport Protocols: Performance Analysis.* PhD thesis, University of Delaware, 1997.