Retransmission-Based Partially Reliable Transport Service: An Analytic Model *

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

Abstract

This paper analytically studies retransmission-based partially reliable transport service. Results show that partially reliable transport service provides increasingly higher throughput and lower delay than reliable transport service as an application's loss tolerance increases and as the underlying network service gets more lossy. Also, to some degree, partially reliable transport service eases the negative effects of ack losses on throughput. Three cost functions associated with the reliability level that a system can support are introduced. These cost functions help demonstrate the penalty when a transport service does not support the ideal reliability level for an application. Results show that the use of reliable transport service when an application only needs a partially reliable transport service can cause considerable throughput drops 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 necessary to avoid the extra cost of reliable transport service, and, at the same time, to guarantee the minimal reliability that an application requires.

1 Introduction

Many applications such as video and audio can tolerate loss. When the network layer provides a best-effort service such as on the Internet, 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 guarantees. This paper investigates partially reliable transport layer protocols that fill the gap between reliable and unreliable transport service by enhancing an unreliable network service *just enough* to allow applications to specify *controlled* levels of loss. 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, and, at the same time, it respects the loss tolerance of the application.

We study the partial reliability guarantees provided through retransmissions by the transport layer. Basically, the transport protocol makes enough retransmissions to provide an expected reliability guarantee to the application. In providing partially reliable service, the transport layer must first *detect* the lost packet and then *decide* whether or not to *recover* it. Depending on the transport entity (i.e., the sender or the receiver) that detects and decides to recover the lost data, two basic techniques in providing partially reliable transport service are possible:

Sender-based loss detection and recovery: The sender has the responsibility of detecting and deciding to recover the 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 receiving transport entity detects the lost data through gap-detection and loss-timers [4]. Once a lost packet is detected and the recovery decision is made, the receiver requests the retransmission of the lost packet by sending negative acks to the sender.

In the literature, different partially reliable services provided by receiver-based techniques have been studied. Application-Oriented Error Control (AOEC) [4] has the objective of satisfying an application's error tolerance with minimum retransmission overhead. In [3], Partially Error-Controlled Connections (PECC) and Slack ARQ are introduced to enable limited recovery of packet losses for stream-based communications in which data completeness must be traded off for low delay service. In AOEC, the lost data is recovered whenever necessary, whereas in PECC and Slack ARQ, the retransmission of the lost data is requested in the transport layer only if it does not delay the application. Although widely rejected, the retransmission of continuous media (e.g., audio) is shown to be feasible by Dempsey in [3] through Slack ARQ. The results of Dempsey are encouraging in terms of providing partially reliable transport

*This work supported, in part, by the National Science Foundation (NCR-9314056), the US Army Communication Electronics Command (CECOM), Ft. Monmouth, and the US Army Research Office (DAAH04-94-G-0093, DAAL03-92-G-0070).

service through retransmissions for multimedia applications. Dempsey mainly uses increased control time[1] to allow timely retransmissions of the lost data.

In this paper, we analytically study partially reliable transport service provided by sender-based loss detection and recovery. Then, we identify the cost of not using the ideal reliability service for an application.[2] This investigation uses a sender-based approach since it provides better performance than a receiver-based approach [6]. The analytic study basically serves the following purposes:

1. To show the performance gains of partially reliable transport service over reliable transport service for applications that can tolerate a certain level of loss, and thus, to motivate the use of partially reliable service against reliable service.

2. To determine the cost of using either more or less reliability than an application needs. These results help show transport service users what penalty they pay by not using the ideal reliability service for their applications. Through these results, the use of partially reliable service against unreliable service is also motivated.

The paper is organized as follows: Section 2 introduces an analytic model and discusses computational results. The cost of not using ideal reliability service for an application is investigated in Section 3. Section 4 summarizes the main results.

2 Analytic Model

We present an analytic model for providing partially reliable transport service using sender-based loss detection and recovery. This analytic model is similar to the one presented in [5] to study the protocol Partial Order Connection (POC).[3] Its major difference is that, in this model, it is unnecessary for all of the objects that are transmitted to eventually be communicated. In [5], no such partial reliability is considered.

The results show that partially reliable transport service provides increasingly better throughput and delay than reliable transport service as the underlying network service gets more lossy and as the application's loss tolerance increases. Later in Section 3 we investigate the penalty paid when there is a mismatch between the application's desired level of reliability and the transport layer's provided level of reliability.

2.1 Introduction to Model

To abstract partially reliable transport service's usage, we use a three layer architecture which includes only the network layer, the transport layer, and the user application layer (see Figure 1). The network layer (called Unreliable NET) is assumed to provide an unreliable service. In 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.

The transport layer enhances the network's unreliable service into a partially reliable service by using sender-based loss detection and recovery. By assumption, the reliability level of an user layer packet is defined by *k_XMIT* reliability as follows:

k_XMIT Reliability: A packet with *k_XMIT* reliability can be transmitted (original plus retransmissions) at most k times. If Transport Sender is still waiting for the ack of a packet after the k^{th} transmission timeout, that packet will be released from Transport Sender's buffers. Releasing a packet from the sender's buffers without receiving an ack for it is called "declaring that packet lost at Transport Sender".

The transport layer provides partially reliable service 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 retransmits the packet if it has not already been transmitted k times. Otherwise, the packet is declared lost. By assumption, there is no problem with running out of buffer space at Transport Receiver.

It is assumed that User Sender submits constant size packets to Transport Sender. It is also assumed that there are infinitely many packets waiting to be communicated at User Sender. User Receiver just accepts packets from Transport Receiver.

The system variables are given in Table 1, and the assumptions about the variables and the system in general are organized in Table 2.

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

The *ideal reliability service* for an application is defined as the reliability level that achieves the best tradeoff between reliability and other QoS parameters to satisfy a given application. Determination of this ideal level of reliability is the responsibility of each specific application (or the user of that application).

2.2 Definitions of Target Values

We analyze the throughput and the delay characteristics of partially reliable transport service and determine the cost of not using ideal reliability service for an application. The throughput and the delay analysis will be done in Section 2.3 by computing the set of target values defined in Table 3. Later in Section 3, the cost functions will be introduced and computed based on the results of Section 2.3.

Figure 2 shows the general model of our system. λ_{US}

[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].

[2] The ideal reliability service for an application is defined in Section 2.1.

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

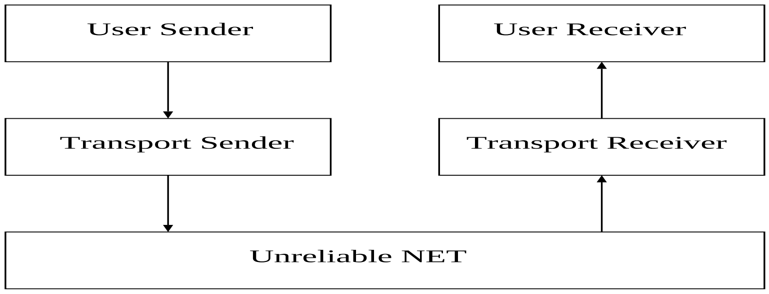


Figure 1: Architecture

System Variables	Definitions
t_{pack}	packet transmission time
t_{delay}	one-way delay for a packet defined as "t_{pack} + one-way propagation delay"
RT	round trip delay defined as "$t_{pack} + 2 *$ one-way propagation 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 Transport Sender
Buf_R	number of buffers at Transport Receiver
p_{succ}	probability of successful packet and ack transmission defined as "$(1-p)*(1-q)$"

Table 1: System variables

and λ_{UR} represent the admission rate at the sender and the throughput at the receiver, respectively. For reliable service (i.e., nothing is permitted to be lost), we have the following equality: $\lambda_{US} = \lambda_{UR} = \lambda_{Reliable}$. In [5], it is shown that $\lambda_{Reliable} = \frac{p_{succ}}{t_{pack}}$. In further sections, we will use this result when comparing partially reliable service with reliable service. The packet loss rate by transport layer (i.e., $\lambda_{loss} = \lambda_{US} - \lambda_{UR}$) depends on the loss probabilities (i.e., p and q) and the levels of reliability that the packets have.

Transport layer delay, *Delay*, is the expected time for a packet to arrive to Transport Receiver, once it is given to Transport Sender. This delay does not include the expected buffering time at Transport Receiver. Through *Delay*, we investigate the delay characteristics of partially reliable service.

Transport Sender Declared Loss Rate, TS_{DLR}, represents the rate at which packets are declared lost at the sender. It is vital to realize that a packet that is delivered at the receiver still might be declared lost at Transport Sender. Such a situation can happen when the ack(s) of the delivered packet is lost. Therefore, we have the following relationship: $\lambda_{UR} + TS_{DLR} \geq \lambda_{US}$ (i.e., the rate of packets that are declared lost at Transport Sender is higher than the rate of the packets that are actually lost by transport layer). For reliable service, both λ_{loss} and TS_{DLR} are zero.

2.3 Computation of Target Values

We will show the performance gains of partially reliable service over reliable service. The results of this section is the basis of Section 3's computations for cost functions. The analysis of this section will proceed as follows: (1) The delivery probability (i.e., PLD) is computed. (2) The computation of admission rate (i.e., λ_{US}) is done by Little's theorem. (3) The throughput (i.e., λ_{UR}) is computed as the product of λ_{US} and PLD. (4) Transport Sender Declared Loss Rate, TS_{DLR}, and Transport layer delay, *Delay*, are computed.

2.3.1 Delivery Probability: PLD

Delivery probability, PLD, is the probability that a packet 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.[4]

$$PLD = 1 - p^k \tag{1}$$

It is noteworthy that PLD is independent of the ack loss rate (i.e., q). Whether we lose *none* or *all* of the acks, delivery probability does not change. Thus, expression (1) shows that *k_XMIT* can provide reliability guarantees regardless of ack loss level. Intuitively, this is because Transport Sender detects and recovers the lost packets without relying on the responses from Transport Receiver. As expected, $PLD = 1-p$ for unreliable service (i.e., $k=1$) and $PLD = 1$ for reliable service (i.e., $k = \infty$). Thus, delivery probability cannot be smaller than the packet success rate (i.e., $1-p$). PLD is virtually one when $k \geq 5$ for all the practical loss levels (e.g., $PLD \geq 1 - 10^{-5}$ when $k \geq 5$ and $p \leq 0.1$). Thus, partially reliable service occurs only when $1 \leq k \leq 5$ for all the practical cases.[5] Expression (1) also shows that allowing few retransmissions (e.g., $k=3$) is sufficient to have high delivery guarantees (e.g., greater than $1-10^{-3}$) at practical loss levels.

[4] The computational details can be found in [6].

[5] The same conclusion can be made through other target values.

	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	There are infinitely many packets waiting to be communicated at User Sender
9	The reliability level for each packet is defined through k_XMIT values

Table 2: Assumptions

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 declared lost per unit time at Transport Sender
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 by 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 at Transport 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 3: Target Values

2.3.2 Throughput: λ_{UR}

Throughput, λ_{UR}, is the rate at which the receiving application (i.e., User Receiver) gets data packets. Some applications may require specific throughput QoS guarantees. This section investigates the conditions for increasing throughput.

Let TS_Time be the expected time that a packet spends at Transport Sender. With assumptions 2, 3, 4, 5, 6, and 8, the number of packets at Transport Sender is always Buf_S. Then by using Little's theorem and the expression for TS_Time, λ_{US} can be computed. λ_{UR} is the product of λ_{US} and PLD:

$$TS\_Time = \frac{1-(1-p_{succ})^k}{p_{succ}} * t_{out} \quad (2)$$

$$\lambda_{US} = \frac{p_{succ}}{t_{pack}} * \frac{1}{1-(1-p_{succ})^k} \quad (3)$$

$$\lambda_{UR} = \frac{p_{succ}}{t_{pack}} * \frac{1-p^k}{1-(1-p_{succ})^k} \quad (4)$$

In expression (4), $\lambda_{UR} > \lambda_{Reliable}$ since $p < (1-p_{succ})$ as long as $q \neq 0$. This expression shows that partially reliable service provides throughput improvements over reliable service as long as there are ack losses in the network layer (i.e., $p < (1-p_{succ})$). Intuitively, this can be explained as follows: since each packet can be transmitted at most k times, after the k^{th} transmission, the unnecessary retransmissions of the packets due to ack losses are avoided. 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 is bounded by $\frac{(1-p)*q}{t_{pack}}$. For 10% network loss level (i.e., $p = q = 0.1$) and 1% application loss tolerance (i.e., $k = 2$), the throughput improvement of partially reliable service over reliable service can be as high as 3%. Figure 3.A shows the relationship between λ_{UR} and k_XMIT values. In the figure, the corresponding reliable service throughput is also given. As shown in the graph, λ_{UR} decreases exponentially as k increases and converges to $\lambda_{Reliable}$.

The relationship between λ_{UR} and loss probabilities is investigated in Figure 3.B. This figure illustrates "λ_{UR} vs p and q" as well as corresponding reliable service throughput. Packet and ack losses have different effects on throughput. In general, λ_{UR} decreases as both p and q increase, but the decrease in λ_{UR} is slower with increasing q than with increasing p. Thus, ack losses are not as detrimental to throughput as packet losses in partially reliable service. As Figure 3.B shows, the gain in throughput over reliable service is small with increasing packet losses. On the other hand, increasing ack loss rate provides more significant throughput advantages over reliable service. Thus, a key result is that partially reliable service mainly provides throughput improvement over reliable service when the network loses acks. One can also make this observation through Figure 3.A. This figure illustrates "λ_{UR} vs k" for $p = 0.1, q = 0.2$ and $p = 0.2, q = 0.1$. The corresponding reliable service throughput is 0.72 packets/unit time for both cases, while the throughput of partially reliable service is higher in the case of higher ack loss rate than in the case of higher packet loss rate. To some degree, partially reliable service overcomes the negative effects of ack losses on the throughput.

Based on these observations, one can say that if network loses packets and acks, and if the application has a high loss tolerance, then one can use partially reliable transport service and have considerable throughput improvement over reliable

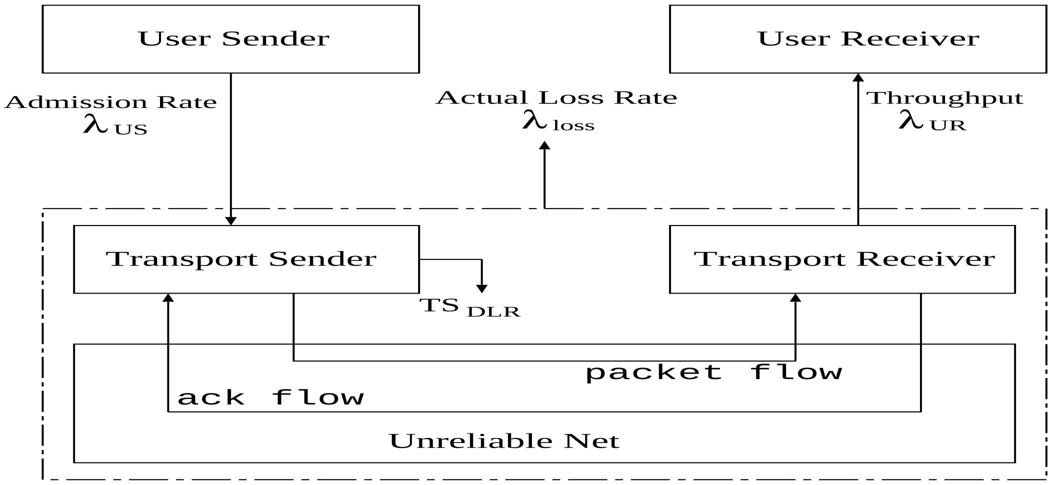
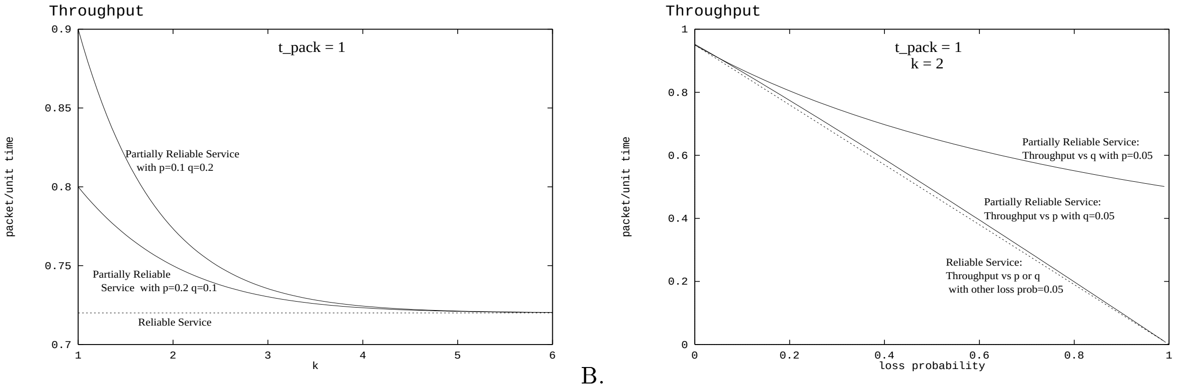


Figure 2: Packet flow rate among different layers



A. B.

Figure 3: Effects of k_XMIT values and loss probabilities on λ_{UR}

transport service.

2.3.3 Transport Sender Declared Loss Rate: TS_{DLR}

We now investigate TS_{DLR}, Transport Sender Declared Loss Rate. TS_{DLR} is the rate at which packets are *declared* lost at Transport Sender. Recall that packets that are declared lost are *not* necessarily the ones undelivered to User Receiver. The investigation of TS_{DLR} is helpful in understanding the overall analytic model.

Let PL be the probability of declaring a packet lost at Transport Sender. $PL = (1 - p_{succ})^k$. TS_{DLR} can be computed as the product of PL and λ_{US}:

$$TS_{DLR} = \frac{p_{succ}}{t_{pack}} * \frac{(1 - p_{succ})^k}{1 - (1 - p_{succ})^k} \quad (5)$$

Expression (5) shows that TS_{DLR} is nonzero for $p_{succ} < 1$. Hence, as long as the network layer loses packets or acks, there will be packets that are declared lost at Transport Sender. This is because, if $p_{succ} < 1$, the probability of not receiving an ack in each of first k transmissions is nonzero, and thus, the probability of not declaring a packet lost is not zero. As an interesting special case, we have the following equality when one assumes acks are never lost (i.e., $q = 0$): $\lambda_{loss} = TS_{DLR} = \frac{1-p}{t_{pack}} * \frac{p^k}{1-p^k}$. Hence, when $q = 0$, $\lambda_{loss} = TS_{DLR}$ (i.e., the packets that are declared lost at Transport Sender are precisely the ones that are not delivered to User Receiver). Thus, $\lambda_{US} = \lambda_{UR} + TS_{DLR}$ for $q = 0$. This can be explained as follows: the packets that are delivered at Transport Receiver will receive their acks since $q = 0$. Hence, it is impossible for a delivered packet to be declared lost at Transport Sender. For $q > 0$, we have $\lambda_{US} > \lambda_{UR} + TS_{DLR}$. This is because, there will be some packets which do arrive at Transport Receiver (and User Receiver) but are declared lost at Transport Sender because of ack losses.

$TS_{DLR} = 0$ for reliable service and $TS_{DLR} = \frac{1-p_{succ}}{t_{pack}}$ for unreliable service. For $k \geq 5$ and $p, q \leq 0.1$, $TS_{DLR} \leq 0.0002 * \lambda_{Reliable}$ (i.e., TS_{DLR} is small as compared to $\lambda_{Reliable}$ when $k \geq 5$ for almost all practical loss levels). When p (or q) is 1, every packet that is given to Transport Sender will be declared lost since no ack will arrive. Thus, when $p = 1$ or $q = 1$, $\lambda_{US} = TS_{DLR} = \frac{1}{t_{pack} * k}$.

2.3.4 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 require specific delay QoS guarantees. For many applications such as real time audio and video, lower delay is even more important than higher throughput. *Delay* does not include the buffering time of the packet at Transport Receiver; it only accounts for the expected time to reach to Transport Receiver for a packet. Once a packet is received, it may remain buffered at Transport Receiver for some time. Generally, lower *Delay* also will cause lower overall delay.

Notice that *Delay* is only computed for the packets that are successfully received by Transport Receiver since the receiving application (i.e., User Receiver) will only experience the delays of such packets. Thus, if a packet is lost k consecutive times, then the transport layer delay for that packet is not included. For $i \leq k-1$, let $P(s_i)$ be the probability that a packet succeeds at $(i+1)^{th}$ try given that the packet is successfully received by Transport Receiver. Then:

$$P(s_i) = P(\text{packet succeeds at } (i+1)^{th} \text{ try} \mid \text{packet succeeds within } k \text{ tries}) = \frac{p^i * (1-p)}{1-p^k} \quad (6)$$

$$Delay = \sum_{i=0}^{k-1} (i * t_{out} + t_{delay}) * P(s_i)$$
$$= t_{delay} + \frac{p + p^k(k*p - k - p)}{1-p} * \frac{t_{out}}{1-p^k} \quad (7)$$

As expected, $Delay = t_{delay}$ for unreliable service and $Delay = t_{delay} + \frac{p}{1-p} * t_{out}$ for reliable service. Thus, potentially, partially reliable service can provide delay improvements as high as $\frac{p}{1-p} * t_{out}$. Figure 4.A shows the relationship between *Delay* and *k_XMIT* values. The corresponding reliable service *Delay* curve also is provided. As the graph shows, partially reliable service provides potentially valuable delay improvement over reliable service even at the practical loss levels. For example, for $p = 0.1$, $t_{out} = 2 * t_{delay}$ and about 1% application loss tolerance (i.e., $k = 2$), *Delay* can be as much as 3.3% lower using partially reliable service. *Delay* increases with increasing k and converges to the delay value of reliable service. Figure 4.B illustrates "*Delay* vs p" for both reliable and partially reliable service. As seen in the graph, the delay gain of partially reliable service over reliable service increases with increasing packet losses.

3 Cost of Not Using Ideal Reliability Service

Traditional computer networks offer either reliable (e.g., TCP) or unreliable (e.g., UDP) transport service. For many applications such as multimedia, neither of these services is ideal because unreliable service lacks any reliability guarantee, while reliable service wastes resources by providing too much reliability. Reliable service does not allow any loss, and thus, the communication system cannot take advantage of an application's loss tolerance. This will result in higher delay and lower throughput than what would have been achievable with ideal reliability service. On the other hand, unreliable service is unable to support the minimal loss guarantees of some applications. Thus, by having to choose either (1) reliable or (2) unreliable service, applications pay a price in the form of either higher delay and lower throughput, or a higher loss rate, respectively. In this section, we investigate the cost of using a transport service that provides either less or more reliability than an application requires.

The notation used in the computation of cost functions is introduced in Table 4. In an ideal case, the application uses a communication system that perfectly supports its desired level of reliability (i.e., $k_{NET} = k_{ideal}$). There is no reliability cost (i.e., penalty) associated with such a case. A cost (or penalty) occurs when $k_{NET} \neq k_{ideal}$: the communication system provides either more or less reliability than the application needs. In Section 2.3, we have shown that with increasing reliability level, throughput decreases and the delay increases. Thus, if a system *NET* provides more reliability than an application needs, such a reliability mismatch will result in lower throughput and higher delay than with the ideal reliability level. In such a case, the application is penalized by lower throughput and higher delay. On the other hand, Section 2.3's results also show that if the communication system lacks reliability guarantees that an application needs, then the application will be penalized by a higher loss level than it could tolerate. Thus, from the results of Section 2.3, we identify three cost functions: (1) throughput, and (2) delay costs of more reliability, and (3) loss cost of less reliability. These cost functions are formally defined in Table 5.

3.1 Costs Incurred by Using More Reliability than Needed

The cost of using more reliability than an application needs is defined in terms of two worsening performance metrics: throughput and delay (see Table 5). These two cost values show the percentage worsening in the corresponding performance metric due to excess reliability provided by the communication system. Since these cost functions are defined when the transport layer provides higher reliability than the ideal case, they are always greater than or equal to 0. For $k_{NET} > k_{ideal}$, they have no practical interpretation. If any of these cost values is equal to "c", this cost should be interpreted as "$100 * c$%" worsening in the corresponding performance metric because of more reliability provided by the transport layer. For example, "$\lambda_{URCost} = 0.1$" means "10%" decrease in the throughput, while "$Delay_{Cost} = 0.2$" means "20%" increase in delay, due to more reliability.

3.1.1 Throughput Cost: λ_{URCost}

The throughput cost,[6] λ_{URCost}, of more reliability can easily be computed by using expression (4):

$$\lambda_{UR_{Cost}} = 1 - \frac{1-(1-p_{succ})^{k_{ideal}}}{1-p^{k_{ideal}}} * \frac{1-p^{k_{NET}}}{1-(1-p_{succ})^{k_{NET}}} \quad (8)$$

[6] In [6], it is shown that the throughput cost is equivalent to the bandwidth cost in system *NET*. The bandwidth cost of using more reliability than needed is defined as the extra bandwidth needed to achieve the throughput of ideal reliability service.

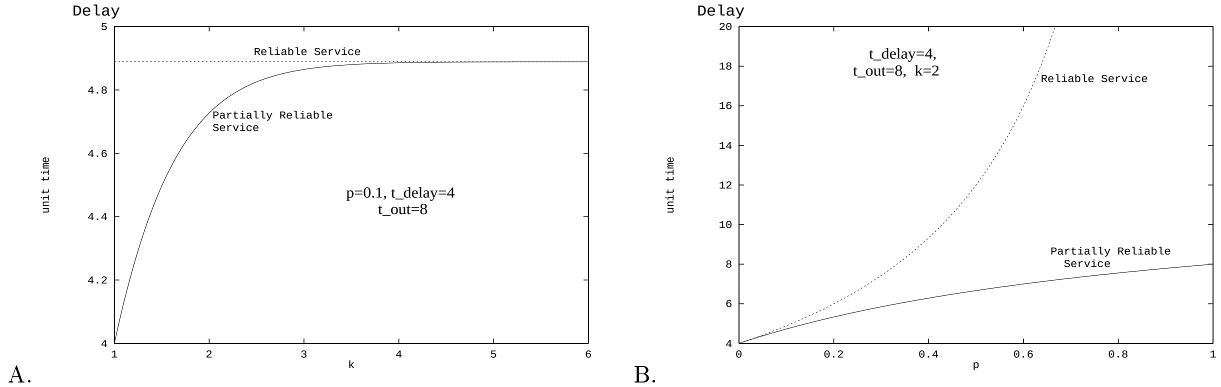


Figure 4: Effects of k_XMIT values and loss probabilities on $Delay$

Notation	Definition
k_{NET}	reliability level that system NET supports
k_{ideal}	ideal reliability level for a given application
PLS	Percentage Loss of partially reliable service in system NET, defined as "$1 - PLD$"
$\lambda_{UR}(k_R)$, $PLS(k_R)$, and $Delay(k_R)$	Throughput, Percentage Loss and $Delay$ in system NET, respectively, with reliability level k_R

Table 4: Notation

The maximal throughput cost that can occur is q since this is the throughput cost of reliable service over unreliable service (i.e., $k_{ideal} = 1$, $k_{NET} = \infty$). For $q = 0.1$, this corresponds to a throughput cost of 0.1 (i.e., 10% decrease in throughput due to unnecessary reliability). Thus, the use of more reliability (e.g., reliable service) can be very costly in terms of throughput. It is worth noting that if ack loss probability is zero, then there is no throughput cost of using more reliability. λ_{URCost} of reliable service over partially reliable service (i.e., $k_{NET} = \infty$ and k_{ideal} is finite) is $\frac{(1-p_{succ})^{k_{ideal}} - p^{k_{ideal}}}{1-p^{k_{ideal}}}$. As expected, for practical loss levels and $k_{ideal} \geq 5$, this cost is negligible. The relationship between λ_{URCost} and k_XMIT is illustrated in Figure 5.A. λ_{URCost} increases and converges to q as the reliability level of the system increases. Figure 5.B plots "λ_{URCost} vs p" and "λ_{URCost} vs q". The throughput cost changes only slightly with increasing packet losses. On the other hand, with increasing ack losses, λ_{URCost} increases steadily and converges to $1 - \frac{k_{ideal}}{k_{NET}}$ as $q \rightarrow 1$. Thus, the throughput cost can be considerably large at high ack loss rates.

The results of this section show that if a network loses acks and the application can tolerate loss, then using more reliability (e.g., reliable service) can result in severe throughput drops. On the other hand, if ack loss probability is low, then there is little throughput penalty in not using the ideal reliability service for the application.

3.1.2 Delay Cost: $Delay_{Cost}$

The delay cost of using more reliability, $Delay_{Cost}$, can be computed by using expression (7) as follows:

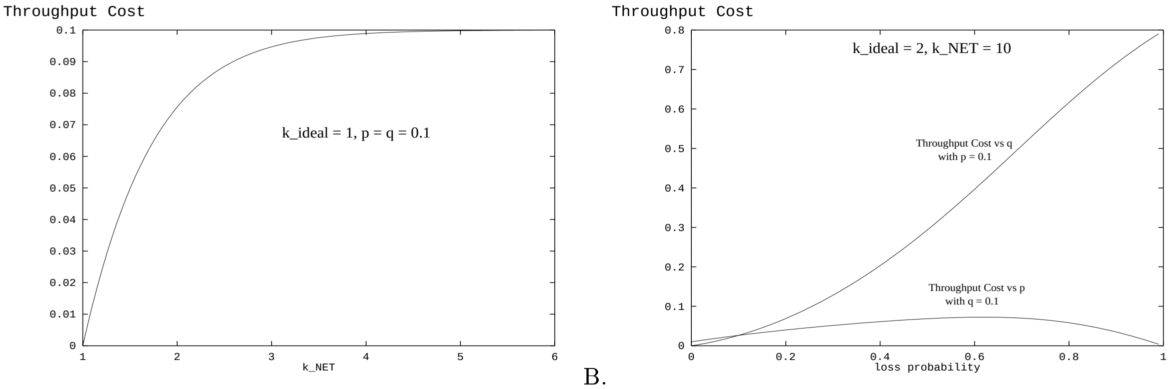
$$Delay_{Cost} = \frac{\left(t_{delay} + \frac{p+p^{k_{NET}}(k_{NET}*p - k_{NET} - p)}{1-p} * \frac{t_{out}}{1-p^{k_{NET}}}\right)}{\left(t_{delay} + \frac{p+p^{k_{ideal}}(k_{ideal}*p - k_{ideal} - p)}{1-p} * \frac{t_{out}}{1-p^{k_{ideal}}}\right)} - 1 \qquad (9)$$

$Delay_{Cost}$ of reliable service over unreliable service is $\frac{\left(\frac{p*t_{out}}{1-p}\right)}{t_{delay}}$. For $t_{out} = 2 * t_{delay}$ and $p = q = 0.1$, this corresponds to a delay cost of 0.222 (i.e., 22.2% increase in delay due to extra reliability). Under the same conditions, the throughput cost is 0.1. Thus, the negative effects of extra reliability (e.g., reliable service) are even worse on delay than on throughput. "$Delay_{Cost}$ vs k_{NET}" is given in Figure 6.A. The delay cost increases as the unnecessary reliability level of the transport layer (i.e., k_{NET}) increases. "$Delay_{Cost}$ vs p" in Figure 6.B shows that $Delay_{Cost}$ increases with increasing packet losses. Thus, the delay cost of using more reliability can potentially be arbitrarily large in lossy networks.

The results of Section 3.1.1 show that the throughput penalty of reliable service increases mainly with increasing ack losses. On the other hand, the results of this section show that unlike λ_{URCost}, the delay cost of reliable service increases with packet losses. Thus, in lossy networks where both packets and acks can be lost, the use of reliable transport service when an application only needs partially reliable transport service can be very costly in terms of both throughput and delay. The applications that can tolerate loss such as audio and video also require certain delay and throughput QoS guarantees. Thus, mainly because of high delay and throughput costs of reliable service (e.g., TCP), such applications often are forced to use unreliable service (e.g., UDP). In the next section we consider the inverse problem: the cost

Cost Function	Definition
Throughput Cost ($\lambda_{UR_{Cost}}$)	Cost of using more reliability in terms of decreased Throughput, defined as $1 - \frac{\lambda_{UR}(k_{NET})}{\lambda_{UR}(k_{ideal})}$ where $k_{NET} \geq k_{ideal}$
Delay Cost ($Delay_{Cost}$)	Cost of using more reliability in terms of increased Transport Layer Delay, defined as $\frac{Delay(k_{NET})}{Delay(k_{ideal})} - 1$ where $k_{NET} \geq k_{ideal}$
Percentage Loss Cost (PLS_{Cost})	Cost of using less reliability in terms of increased Percentage Loss, defined as $PLS(k_{NET}) - PLS(k_{ideal})$ where $k_{NET} \leq k_{ideal}$

Table 5: Cost functions



A. B.

Figure 5: Effects of k_XMIT values and loss probabilities on λ_{URCost}

of using less reliability than an application needs (e.g., unreliable service).

3.2 Loss Cost of Less Reliability

In Sections 3.1, we quantified the cost when applications must choose to use more reliability (e.g., reliable service) than they need. Due to such costs of reliable service and unavailability of partially reliable service, applications may choose to use unreliable service. In such a situation, the application uses less reliability at the expense of losing more packets than it really wants to tolerate. In this section, we investigate the cost of using less reliability.

The results of Section 2.3 show that as the reliability level of the transport layer decreases, percentage loss (PLS) increases. Thus, we will characterize the cost of using less reliability as the increase in percentage loss (i.e., $PLS_{Cost} = PLS(k_{NET}) - PLS(k_{ideal})$). PLS_{Cost} is defined only when the system NET provides less reliability than the ideal case (i.e., $k_{NET} \leq k_{ideal}$). Thus, this cost value is always greater than or equal to 0. For $k_{NET} > k_{ideal}$, PLS_{Cost} has no practical interpretation. "$PLS_{Cost} = c$" should be interpreted as "$100 * c\%$" more loss than the application can tolerate.

3.2.1 Percentage Loss Cost: PLS_{Cost}

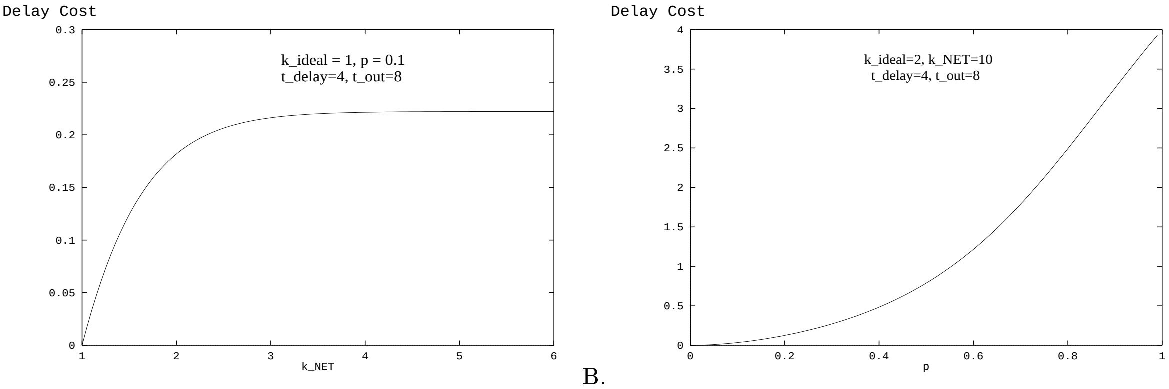
The percentage loss cost of less reliability, PLS_{Cost}, can be computed by using expression (1) as follows:

$$PLS_{Cost} \quad = \quad p^{k_{NET}} - p^{k_{ideal}} \tag{10}$$

The percentage loss cost of unreliable service over reliable service (i.e., $k_{NET} = 1$, $k_{ideal} = \infty$) is p. Thus, $0 \leq PLS_{Cost} \leq p$. As expected, if packet loss rate of the network layer is negligible, then PLS_{Cost} will also be negligible. "PLS_{Cost} vs k_{ideal}" is given in Figure 7.A. PLS_{Cost} increases as the reliability need of the application (i.e., k_{ideal}) increases. Similarly, "PLS_{Cost} vs p" in Figure 7.B shows that the loss cost of less reliability first increases, then starts decreasing and goes to 0 with increasing p. Thus, in lossy networks this cost can be large.

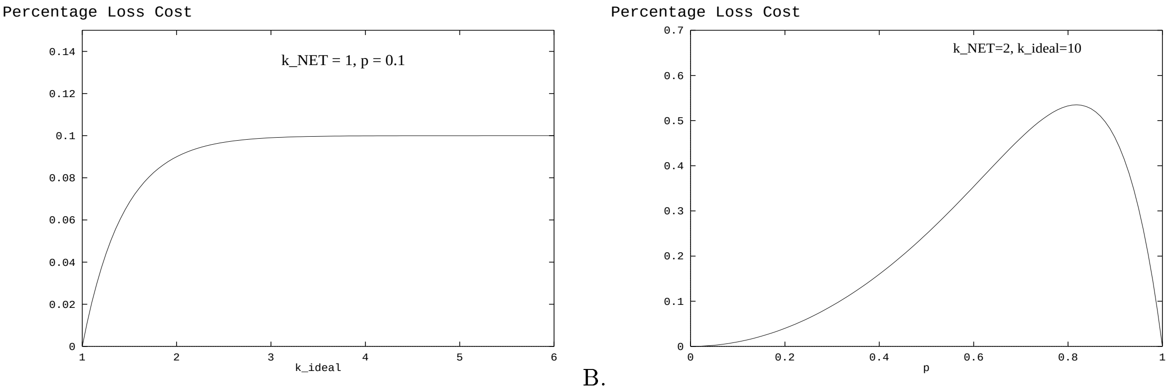
These results show that the loss cost of using less reliability (e.g., unreliable service) only depends on packet loss rate, and this cost cannot be greater than p. In general, if the overall loss tolerance level of an application is higher than the overall packet loss level of the network layer, then such an application can use unreliable service. On the other hand, even though the average loss level of a network is smaller than the overall loss tolerance level of the application, there can be times where the network loses large amounts of data that the application cannot tolerate. This is especially the case with packet-switched networks where the losses mainly occur in bursts due to buffer overflows. Notice that our analytic model assumes random (i.e., Bernoulli) losses, and thus, the cost of burst losses is not studied.

The results of Section 3.1 show that if the transport layer provides higher reliability than needed, this can result in severe throughput drops and delay increases in lossy networks. On the other hand, this section shows that at high loss rates, unreliable service cannot respect the loss tolerance of applications. Thus, in lossy environments, partially reliable service is necessary to avoid the price of reliable service, and at the same time, to provide the reliability guarantees for applica-



A. B.

Figure 6: Effects of k_XMIT values and loss probabilities on $Delay_{Cost}$



A. B.

Figure 7: Effects of k_XMIT values and loss probabilities on PLS_{Cost}

tions.

4 Summary of Main Results

This paper studies partially reliable transport service via an analytic model. We investigate the effects of packet and ack losses, as well as various levels of application loss tolerance on the system performance. The results show that partially reliable transport service provides considerable throughput, admission rate, and delay improvements over reliable transport service when the underlying network service is lossy and an application has a high loss tolerance. It is also shown that, to some degree, partially reliable transport service eases the negative effects of ack losses on throughput.

Three cost functions associated with the level of reliability that the communication system can support are also introduced. The three cost functions are (1) throughput, and (2) delay costs of using more reliability, and (3) loss cost of using less reliability. These functions show the cost of not using ideal reliability service for an application in terms of three worsening performance metrics. The results show that the use of reliable transport service instead of partially reliable transport service can result in considerable throughput drops and delay increases in lossy networks. On the other hand, at high loss levels, unreliable transport service is unable to support an application's minimal loss guarantee. Thus, in lossy environments, partially reliable transport service is necessary to avoid the price of reliable transport service, and at the same time, to provide reliability guarantees for applications.

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

[3] Bert J. Dempsey. *Retransmission-Based Error Control For Continuous Media Traffic In Packet-Swithced Networks.* PhD Dissertation, 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, P. T. Conrad, and G. Burch. Partial Order Transport Service: An Analytic Model. In 9^{th} *Annual IEEE Workshop on Computer Communications*, Marathon, Florida, October 1994. IEEE.

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