

Performance of Counting Protocols for Reliable End-to-End Sequence Transmission*

(Extended Abstract)

Richard E. Ladner
Anthony LaMarca
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
U.S.A.

Ewan Tempero
Department of Computer Science
Victoria University of Wellington
P.O. Box 600
Wellington
NEW ZEALAND

Abstract

We introduce and evaluate the performance of several new protocols from a new class of protocols, called *counting protocols*. Counting protocols provide a reliable FIFO channel in computer networks in which packets may be lost or delivered out of order.

Three counting protocols are presented: (i) the *one-bit protocol* which uses one bit header and sends one packet per message under ideal conditions, but performs poorly in networks with realistic loss rates, (ii) the *mode protocol* which uses multiple-bit headers and behaves well in networks with a realistic loss rates, and (iii) the *mode-power protocol* which also uses multiple-bit headers and performs better than the mode protocol on short sequences, but worse on very long sequences.

We do a careful performance analysis of both the one-bit and mode protocols and present the results of a simulation study to demonstrate the performance of the mode-power protocol.

*This research was supported by NSF Grant No. CCR-9108314. LaMarca was supported in part by an AT&T scholarship. Corresponding author is Richard Ladner, ladner@cs.washington.edu

1 Introduction

In this paper we introduce and evaluate the performance of new protocols to solve the *sequence transmission problem* (STP), which is an abstraction of the classic computer network problem of providing a reliable virtual circuit service on top of an underlying datagram service [Tan88, Sta91]. A virtual circuit service guarantees that an input sequence of messages originating at the service at one site in the network is delivered in order, without loss or duplication, at another site. A datagram service attempts to deliver an input sequence of messages at one site to another site but makes no guarantees about delivery of packets except not to mutate or duplicate. TCP is a standard example of a virtual circuit. TCP is built on top of IP which provides the functionality of a datagram service [Pos81a, Pos81b]. A protocol for STP, such a TCP, is often called an *end-to-end* protocol because it is only implemented on the remote sites and relies on an underlying network protocol to provide the necessary services for full communication.

The standard approach to solving STP is for the sending process to include the *sequence number* as part of the header to a packet containing the n -th message in the input sequence. The receiving process will be able to reconstruct the sequence order from the sequence numbers provided that the protocol provides adequate error control to resend packets that may have been lost. Since packets, and hence their headers, must have bounded length, practical protocols use *bounded sequence numbers*. For example, TCP uses 32 bit sequence numbers [Pos81b]. Traditional bounded sequence number protocols alone do not provide reliable communication in the presence of arbitrarily reordered packets. For this reason, the underlying network protocol IP uses the “hop count” mechanism to purge old packets from the network, thereby allowing TCP to use bounded sequence numbers and provide reliable communication [Pos81a]. Without a method of guaranteeing the removal of old packets from the network, bounded sequence numbers could wrap around, thereby allowing a bounded sequence number protocol to fail.

Our protocols to solve STP are based on theoretical protocols for STP developed by Attiya *et al.* [AFWZ89] and improved on by Afek *et al.* [AAF⁺91], Tempero and Ladner [TL90], and Tempero [Tem90]. A main feature of these and our protocols is that they do *not* use the technique of sequence numbers to correctly maintain the order of a sequence of messages. Rather, they use a new counting technique for maintaining the correct sequence order. We call this new class of protocols *counting protocols*. Most surprising is that counting protocols use bounded size headers and still maintain correct sequence order even if the underlying network protocol never discards old packets.

1.1 The Model

To be more specific about our model, we consider two asynchronous processes, the *Sender* and the *Receiver*, which communicate with each other over a *channel*. The Sender has an infinite sequence of *messages*, $x_1, x_2, x_3 \dots, x_n, \dots$ that it must transmit to the Receiver. The Sender and Receiver may each send and receive *packets*, each of which consists of a header and optionally a message. Both the Sender and Receiver have a timeout mechanism which eliminates the possibility of waiting forever for the receipt of a packet that may be lost. The sequence transmission problem (STP) is the problem of designing a protocol for the Sender and Receiver which results in the Receiver eventually writing each member of the input sequence in order.

We identify four types of channels:

1. *Non-FIFO Channel* may lose packets or deliver them in any order. A packet that is sent may be delivered at any time in the future, thus it is impossible for the Sender or Receiver to know if a packet is lost or simply being held in the channel.
2. *FIFO Channel* may lose, but not reorder packets.
3. *Statistical FIFO Channel* is a FIFO channel that loses packets at a fixed rate p .
4. *Ideal Channel* is a statistical FIFO channel with loss rate zero.

Although our protocols are correct in an adversarial non-FIFO channel, for simplicity we will evaluate the performance of our protocols in a statistical FIFO channel.

1.2 Results

The original counting protocols are far from practical. Indeed, Mansour and Schieber showed that any STP protocol that uses bounded headers is expected to send α^n packets to deliver the n -th message in a statistical FIFO channel with positive loss rate [MS89]. The base α depends on the header size and packet loss rate. Thus, it would appear that there is no hope for an efficient counting protocol. In this paper we demonstrate that efficient counting protocols are possible in spite of this exponential behavior. All our protocols are correct in a non-FIFO channel, but their performance is analyzed in a statistical FIFO channel.

We develop a progression of three new counting protocols for solving STP:

1. *One-bit protocol*, P_{one} , is a counting protocol that sends exactly one packet per message in an ideal channel, but behaves poorly in a statistical FIFO channel with a realistic loss rate. P_{one} uses only one bit of header. This counting protocol is the first to exhibit the perfect behavior of one packet per message in an ideal network. The analysis of P_{one} is quite interesting. We prove that the expected number of packets needed to send the n -th message in a statistical FIFO channel with loss rate $p > 0$ is $\Theta(\alpha^n)$ where

$$\alpha = \frac{1}{(1-p)^2} - \frac{1}{2} + \sqrt{\frac{1}{(1-p)^2} - \frac{3}{4}}.$$

2. *Mode protocol*, P_{mode} , is a counting protocol that uses multiple-bit headers and behaves well in a statistical FIFO channel with a realistic loss rate. For realistic loss rates, relatively small headers can be used to make the exponent base α small enough to be useful. We prove that the expected number of packets needed to send the n -th message in a statistical FIFO channel with loss rate $p > 0$ is α^n where

$$\alpha = \sqrt[b]{\frac{2}{(1-p)^2} - 1},$$

and where b is the number of bits of header used by the protocol.

3. *Mode-power protocol*, P_{mp} , is a counting protocol that also uses multiple-bit headers and also behaves well in a statistical FIFO channel with a realistic loss rate. P_{mp} behaves better than P_{mode} in the short term, but worse in the long term. The header in P_{mp} is divided into two

fields, power and mode. Given a fixed header size, the performance of P_{mp} depends how many bits are assigned to each field. The performance of P_{mode} is evaluated using a simulation study.

We define the *packet utilization* of a protocol to be the expected number of messages per packet to send the first n messages. Thus, packet utilization is always between zero and one, with one being a perfect one message per packet. We use packet utilization as the performance analysis tool to evaluate and compare our protocols.

A complete description of all the protocols, proof of correctness, and analysis can be found in our unpublished technical report [LLT92].

1.3 Related Work

There is a vast amount of literature on end-to-end protocols for sequence transmission and several text books cover the basics well [Tan88, Sta91]. It was unknown until recently whether or not the sequence transmission problem was solvable using bounded headers in a non-FIFO channel. Indeed, the early evidence indicated that no such protocol existed. Lynch, Mansour, and Fekete showed that any bounded header protocol must use an unbounded number of packets per message in a non-FIFO channel [LMF88]. Surprisingly, Attiya *et al.*, using a counting protocol, showed that STP was solvable using bounded headers in a non-FIFO channel [AFWZ89]. This work was later extended by Afek *et al.* [AAF⁺91]. Tempero and Ladner refined the counting protocol technique [Tem90, TL90]. Mansour and Schieber showed that any sequence transmission protocol for non-FIFO channels that uses bounded headers requires α^n packets on average to send the first n messages on a statistical FIFO channel, for some α that depends on the loss rate and header size [MS89].

Other theoretical work on the sequence transmission problem is concerned with upper and lower bounds on the number of packets required to send the n -th message for protocols in an adversarial non-FIFO channel [LMF88, MS89, WZ89, TL90, AFWZ89, AAF⁺91]. One basic result of Tempero and Ladner is that there is a counting protocol with the property that no matter how poorly the channel behaves in delivering the first $n - 1$ messages, if the channel subsequently behaves well then the n -th message can be delivered using $O(n)$ packets [TL90]. This protocol uses two modes, a primary mode that uses at most a linear number of packets and a backup mode that resembles our P_{one} . This two-mode protocol inspired our use of multiple modes in P_{mode} .

2 The One-Bit Protocol

Although the one-bit protocol, P_{one} , does not have practical significance we introduce it for two reasons. First, we use it as a vehicle for explaining the class of counting protocols. Second, it has theoretical significance because it is the first protocol that solves STP in a non-FIFO channel and sends exactly one packet per message in an ideal channel. Our protocol P_{one} , given in Figures 1 and 2, is derived from a receiver driven version of the well known solution to STP in a FIFO channel known as the *alternating bit protocol* (ABP) [BSW69] and the original counting protocols [AFWZ89, AAF⁺91].

The basic idea in P_{one} and other counting protocols is for the Sender and Receiver to maintain counters u_S and u_R , respectively, which bound the number of packets which could be in the channel. The Receiver is convinced that a message has been delivered if it receives more than u_R packets

containing the message. The Sender is convinced to move on to the next message when it has received at least u_S packets containing a request for the next message. Several new ideas must be employed in order to achieve the perfect one packet per message performance in an ideal channel.

Although P_{one} is designed to be correct on a non-FIFO channel it is very difficult to analyze in such a general setting. Instead, we choose to analyze the protocol in a much simpler setting, namely on a statistical FIFO channel, where each packet has a chance p of being lost. We have added the additional constraint that the protocol's timeout value is large enough that timeouts only occur when packets are lost.

The performance of the protocol can be measured by seeing how many packets the Sender and Receiver send and receive per message. Let:

- $Q_{sR}(n)$ be the expected number of packets sent by the Receiver during **ReceiveMsg**(n).
- $Q_{rR}(n)$ be the expected number of packets received by the Receiver during **ReceiveMsg**(n).
- $Q_{sS}(n)$ be the expected number of packets sent by the Sender during **SendMsg**(n).
- $Q_{rS}(n)$ be the expected number of packets received by the Sender during **SendMsg**(n).

Alternatively, one might be interested in time per message, rather than packets per message. Since the protocol is receiver-driven, $Q_{sR}(n) - Q_{rR}(n)$ is number of packets that have been lost. If we define T to be a fixed timeout value and R to be packet round trip time, the expected elapsed time for the protocol is $T(Q_{sR}(n) - Q_{rR}(n)) + R(Q_{rR}(n))$.

A simple way to evaluate our protocols' performance is in terms of *packet utilization*, which is defined as the expected number of messages per packet sent by the Sender for the first n messages. In the case of P_{one} , packet utilization is defined to be $n / \sum_{i=1}^n Q_{sS}(i)$. This quantity gives us an idea of the utilization achieved by the protocol, where perfect utilization is exactly one message per packet.

The expected number of packets sent or received is intimately related to the expected size of the Sender's and Receiver's counters. Since u_S is ambiguous with respect to time, for $n \geq 0$, define $u_S(n)$ to be the value of u_S at the time of the call to **SendMsg**($n + 1$). Similarly, let $u_R(n)$ be the value of u_R at the time of the call to **ReceiveMsg**($n + 1$). Define $\bar{u}_S(n)$ and $\bar{u}_R(n)$ to be the expected value of $u_S(n)$ and $u_R(n)$ respectively over all runs of the protocol.

Theorem 1. *The following equations define the expected number of packets sent and received in P_{one} in terms of the expected values of the counters. For all $n \geq 1$:*

$$\begin{aligned}
Q_{rR}(1) &= 1 \\
Q_{rR}(n) &= \frac{1}{1-p} \bar{u}_S(n-2) + \bar{u}_R(n-1) + \frac{1}{1-p}, \quad n \geq 2 \\
Q_{sR}(n) &= \frac{1}{(1-p)^2} Q_{rR}(n) \\
Q_{rS}(n) &= \frac{1}{1-p} \bar{u}_R(n-1) + \frac{1}{(1-p)^2} \bar{u}_S(n-1) + \frac{1}{(1-p)^2} \\
Q_{sS}(n) &= Q_{rS}(n)
\end{aligned}$$

These formulas are derived in the full paper.

These formulas tells us two things. First, the performance of the protocol is tied directly to the value of the counters. Second, that we need to develop expressions for the expected value of the counters if we want to do any further performance analysis. Analysis of the protocol demonstrates that the expected values of the counters can be defined by a set of recurrences.

Theorem 2. *The values of $\bar{u}_R(n)$ and $\bar{u}_S(n)$ satisfy the following recurrences:*

$$\begin{aligned}\bar{u}_R(n) &= \frac{1}{(1-p)^2}\bar{u}_R(n-1) + \frac{1-(1-p)^2}{(1-p)^3}\bar{u}_S(n-2) + \frac{1-(1-p)^2}{(1-p)^3}, & n \geq 2 \\ \bar{u}_S(n) &= \frac{1}{(1-p)^2}\bar{u}_S(n-1) + \frac{1-(1-p)^2}{(1-p)}\bar{u}_R(n-1) + \frac{1-(1-p)^2}{(1-p)^2}, & n \geq 1\end{aligned}$$

The initial conditions are $\bar{u}_R(0) = 0$, $\bar{u}_R(1) = \frac{1-(1-p)^2}{(1-p)^2}$, and $\bar{u}_S(0) = 0$.

We can now evaluate how well P_{one} performs on an ideal channel. If we substitute 0 for p in the recurrences, they reduce to $\bar{u}_S(n) = \bar{u}_R(n) = 0$ for $n \geq 0$ and $Q_{sS}(n) = Q_{rS}(n) = Q_{sR}(n) = Q_{rR}(n) = 1$ for $n \geq 1$. If the channel always behaves perfectly, the counters never grow and P_{one} achieves an optimal one packet per message.

If the channel has a loss rate $p > 0$ then the recurrences for $\bar{u}_R(n)$ and $\bar{u}_S(n)$ can be solved using the method of generating functions [GK82]. The exact solution is very long expression. It suffices to say that $\bar{u}_R(n)$ and $\bar{u}_S(n)$ are both $\Theta(\alpha^n)$ ¹ where

$$\alpha = \frac{1}{(1-p)^2} - \frac{1}{2} + \sqrt{\frac{1}{(1-p)^2} - \frac{3}{4}}.$$

As would be expected α is just slightly larger than $\frac{1}{(1-p)^2}$. It is clear from our analysis that packet utilization goes to zero at a rate proportional to $n\alpha^{-n}$. Since α is a function of the error rate p , it is interesting to see how the packet utilization is affected by the loss rate. Figure 3 shows the performance of P_{one} measured in messages per packet for several values of $p > 0$. For reasonable error rates such as .001 and sequence lengths 1000, P_{one} achieves a packet utilization of less than 10%. Thus, although P_{one} is correct, but its performance is far from practical.

3 The Mode Protocol

While the P_{one} protocol is correct in a non-FIFO channel, it performs poorly in a statistical FIFO channel. The inefficiency in P_{one} comes from the compounding effect that lost packets have. A lost packet results in the incrementing of the processes counter. This in turn results in more packets being sent per message which increases the expected number of lost packets per message. Ideally we would like to add a mechanism that would slow the growth of the processes counters. The rate of growth, however, is determined by the behavior of the channel, and cannot be changed without changing the semantics of the counter.

Packets in P_{mode} are P_{one} packets with the addition of an b bit field to the headers. We call this the *mode* field, and it allows for $m \leq 2^b$ modes, although P_{mode} need not use all the modes possible

¹A function $f(n)$ is $\Theta(g(n))$ if there are positive constants c and d such that $cg(n) \leq f(n) \leq dg(n)$ for all but finitely many n

with b bits. Since packets with different modes are clearly distinguishable, the protocol no longer needs to maintain a single counter. Instead, it can maintain a single counter per mode. Rather than maintaining the counters u_S and u_R , the processes will maintain counter vectors $u_S[0..m-1]$ and $u_R[0..m-1]$. The addition of modes also offers the advantage that packets lost in a particular mode minimally affect the performance of messages sent in a different mode. In order to utilize all of the modes, message n is sent in mode $n \bmod m$.

The analysis of the performance P_{mode} has many similarities to the analysis of P_{one} . We define Q_{sR} , Q_{rR} , Q_{sS} , and Q_{rS} just as before, indicating the expected number of packets sent and received by the Receiver and Sender respectively. As before, the expected number of packets sent or received is intimately related to the expected size of the Sender's and Receiver's counters. In this case each of the Receiver and Sender have m counters where m is the number of modes. For $0 \leq i < m$ and for $n \geq 0$, define $u_S[i](n)$ to be the value of $u_S[i]$ at the time of the call to **SendMsg**($n+1$). Similarly, let $u_R[i](n)$ be the value of $u_R[i]$ at the time of the call to **ReceiveMsg**($n+1$). Define $\bar{u}_S[i](n)$ and $\bar{u}_R[i](n)$ to be the expected value of $u_S[i](n)$ and $u_R[i](n)$ respectively over all runs of the protocol.

Theorem 3. *The following equations define the expected number of packets sent and received in P_{mode} in terms of the expected values of the counters. For all $n \geq 1$: For all $n \geq 1$:*

$$\begin{aligned} Q_{rR}(1) &= 1 \\ Q_{rR}(n) &= \frac{1}{1-p} \bar{u}_S[n \bmod m](n-2) + \bar{u}_R[n \bmod m](n-1) + \frac{1}{1-p}, \quad n \geq 2 \\ Q_{sR}(n) &= \frac{1}{(1-p)^2} Q_{rR}(n) \\ Q_{rS}(n) &= \frac{1}{1-p} \bar{u}_R[n \bmod m](n-1) + \frac{1}{(1-p)^2} \bar{u}_S[n+1 \bmod m](n-1) + \frac{1}{(1-p)^2} \\ Q_{sS}(n) &= Q_{rS}(n) \end{aligned}$$

If $m = 1$ then the analysis of P_{mode} is identical to that of P_{one} . If $m \geq 2$ then the expected values of the counters are given by a similar set of recurrences.

Theorem 4. *The values of $\bar{u}_R[i](n)$ and $\bar{u}_S[i](n)$ satisfy the following recurrences:*

$$\bar{u}_R[i](n) = \begin{cases} \frac{1}{(1-p)^2} \bar{u}_R[i](n-1) + \frac{1-(1-p)^2}{(1-p)^3} \bar{u}_S[i](n-2) + \frac{1-(1-p)^2}{(1-p)^3}, & i \equiv n \bmod m \\ \bar{u}_R[i](n-1), & i \not\equiv n \bmod m \end{cases}$$

$$\bar{u}_S[i](n) = \begin{cases} \bar{u}_S[i](n-1) + \frac{1-(1-p)^2}{(1-p)} \bar{u}_R[i](n-1) + \frac{p}{1-p}, & i \equiv n \bmod m \\ \frac{1}{(1-p)^2} \bar{u}_S[i](n-1) + \frac{p}{(1-p)^2}, & i \equiv n+1 \bmod m \\ \bar{u}_S[i](n-1), & i \not\equiv n \bmod m \text{ and } i \not\equiv n+1 \bmod m \end{cases}$$

The recurrence for $\bar{u}_R[i](n)$ holds for all $n \geq 2$ and the recurrence for $\bar{u}_S[i](n)$ holds for all $n \geq 1$. The initial conditions are $\bar{u}_R[i](0) = 0$, $\bar{u}_R[i](1) = \frac{1-(1-p)^2}{(1-p)^2}$, and $\bar{u}_S[i](0) = 0$ for $0 \leq i \leq m-1$.

Surprisingly, these recurrences can be solved almost exactly. To begin with the counter recurrences can be simplified by defining three new quantities:

$$\begin{aligned} R(n) &= \bar{u}_R[n \bmod m](n) \\ D(n) &= \bar{u}_S[n \bmod m](n) \\ C(n) &= \bar{u}_S[n+1 \bmod m](n) \end{aligned}$$

The expected counter values can be recovered from the values of R , D , and C using the equations:

$$\begin{aligned}\bar{u}_R[i](n) &= R(n - ((n - i) \bmod m)) \\ \bar{u}_S[i](n) &= \begin{cases} D(n - ((n - i) \bmod m)), & i \not\equiv n + 1 \pmod m \\ C(n), & i \equiv n + 1 \pmod m \end{cases}\end{aligned}$$

We use the convention that $R(n) = D(n) = C(n) = 0$ if $n \leq 0$.

The values of $R(n)$, $D(n)$, and $C(n)$ can be expressed with the recurrences:

$$\begin{aligned}R(n) &= \frac{1}{(1-p)^2}R(n-m) + \frac{1-(1-p)^2}{(1-p)^3}D(n-m) + \frac{1-(1-p)^2}{(1-p)^3}, \quad n \geq 2 \\ D(n) &= C(n-1) + \frac{1-(1-p)^2}{1-p}R(n-m) + \frac{p}{1-p}, \quad n \geq 1 \\ C(n) &= \frac{1}{(1-p)^2}D(n-m+1) + \frac{p}{(1-p)^2}, \quad n \geq 1\end{aligned}$$

The initial conditions are $R(n) = D(n) = C(n) = 0$ for $n \leq 0$ and $R(1) = \frac{1-(1-p)^2}{(1-p)^2}$. These three recurrences can immediately be reduced to two by substituting the expression for $C(n-1)$ into the expression for $D(n)$, obtaining the equation

$$D(n) = \frac{1}{(1-p)^2}D(n-m) + \frac{1-(1-p)^2}{1-p}R(n-m) + \frac{1-(1-p)^2}{(1-p)^2}, \quad n \geq 1$$

The values of $R(n)$ and $D(n)$ can be closely bounded by simple exponential forms:

Theorem 5. For all $n \geq 0$:

$$\begin{aligned}\frac{1}{2(1-p)}\alpha_m^n - \frac{1}{2(1-p)} &\leq R(n) \leq \frac{2-(1-p)^2}{2(1-p)^3}\alpha_m^n - \frac{1}{2(1-p)} \\ \frac{1}{2}\alpha_m^n - \frac{1}{2} &\leq D(n) \leq \frac{2-(1-p)^2}{2(1-p)^2}\alpha_m^n - \frac{1}{2}\end{aligned}$$

where

$$\alpha_m = \sqrt[m]{\frac{2}{(1-p)^2} - 1}.$$

Thus, we have the following bound.

Theorem 6. For small p and $n \geq 0^2$: $R(n) \sim D(n) \sim \frac{1}{2}\alpha_m^n - \frac{1}{2}$.

Returning to the performance of P_{mode} , the equations for the expected number of packets sent or received by the Sender and Receiver can be rewritten in terms of the functions R and D as

²If $F_p(n)$ and $G_p(n)$ are two non-negative valued function parameterized by p then we say that $F_p(n) \sim G_p(n)$ if for all $\epsilon > 0$ there is $P > 0$ such that for all $p < P$ and n , $(1-\epsilon)G_p(n) \leq F_p(n) \leq (1+\epsilon)G_p(n)$. The relation \sim is an equivalence relation on functions parameterized by p .

follows:

$$\begin{aligned}
Q_{rR}(1) &= 1 \\
Q_{rR}(n) &= \frac{1}{1-p}D(n-m) + R(n-m) + \frac{1}{1-p}, \quad n \geq 2 \\
Q_{sR}(n) &= \frac{1}{(1-p)^2}Q_{rR}(n) \\
Q_{rS}(n) &= \frac{1}{1-p}R(n-m) + \frac{1}{(1-p)^2}D(n-m+1) + \frac{1}{(1-p)^2} \\
Q_{sS}(n) &= Q_{rS}(n)
\end{aligned}$$

Thus, we may conclude:

Theorem 7. For small p and $n \geq 1$: $Q_{rR}(n) \sim Q_{sR}(n) \sim Q_{rS}(n) \sim Q_{sS}(n) \sim \alpha_m^n$.

It is interesting to evaluate the effect of header size on the packet utilization, $n / \sum_{i=1}^n Q_{sS}(i)$. For small p and any $m \geq 2$ the packet utilization declines to zero at the rate approximately equal to $U_m(n) = \frac{(\alpha_m - 1)n}{\alpha_m^{n+1} - \alpha_m}$, where n is the message number. Using the fact that $\alpha_{2m} = \sqrt{\alpha_m}$, we can argue that

$$U_{2m}(2n) = \frac{2}{1 + \frac{1}{\alpha_{2m}}} U_m(n).$$

Thus, $U_{2m}(2n)$ is just slightly larger than $U_m(n)$. The effect is that for a given acceptable packet utilization, one extra bit of header allows twice as many messages to be transmitted than can without the extra bit of header.

Figure 4 shows the packet utilization of P_{mode} for a fixed loss rate and varying header size. The dramatic improvement in packet utilization with growing header size can be readily observed.

Figure 5 shows how many messages can be sent at a fixed loss rate, a fixed packet utilization, and a varying header size. As expected, the length of sequence that can be transmitted doubles with each bit added to the header. As an example from figure 5, with a loss rate of .001, packet utilization 90%, and 10 bits of header the number of messages that can be sent is more than 10,000.

4 The Mode-Power Protocol

The P_{mode} protocol uses additional bits in the header to reduce the base of the exponent in the counter growth expression. This provides for a long term performance improvement over P_{one} . Our second modification provides a temporary but more profound performance improvement for our STP protocols. Our second idea focuses on the source of the protocols' inefficiency, namely the need to send multiple copies of a single packet.

To alleviate this problem, we propose the idea of having packets of varying denominations similar to a monetary system. With packets of multiple denominations, fewer packets need to be sent per message. For example, if originally eleven packets needed to be received, it would now suffice to receive two packets of denomination five and one packet of denomination one. In order to implement this idea, we added a j bit *power* field to the header. The *power* field denotes a packet's denomination and allows us to have packets with power from 1 to 2^j . This modification

offers a substantial performance improvement by allowing the processes to complete a single packet-cycle per message while the counters are less than or equal to 2^j . Unfortunately the benefits are temporary in that once the counters exceed 2^j , packets per message grows at the previous rate. Our most efficient protocol, P_{mp} , resulted from adding this power mechanism to P_{mode} .

We have not developed an expression for the performance of the P_{mp} protocol in a statistical FIFO channel. This is largely due to the complexity of the protocol, specifically the existence of packets with varying power. Instead we rely on simulation data to illustrate points about the performance of P_{mp} .

In P_{mp} , the power field and the mode field provide different performance benefits. The mode field provides long term performance gain by reducing the base of the exponent in the counter growth expression, while the power field provides very efficient performance for a short period of time.

Figure 6 demonstrates these two effects. This graph represents simulation data for the P_{mp} protocol using a packet loss rate of .005 and five bits of header. Recall that the protocol uses one bit for the sequence number, leaving four bits to be used by the power and mode fields. One curve represents performance when all four bits are used for the power field, while the other curve represents performance when all four bits are used for the mode field. As expected, the all-power curve performs better in the initial stages and then degrades quickly. Different partitionings of the bits between power and mode result in a blending of these two curves.

This graph brings up an important question: Given that we will be solving STP using the P_{mp} protocol with b bits of header, how should the $b - 1$ bits be partitioned between the mode field and the power field to provide optimal performance?

Figure 7 shows simulation data for P_{mp} running with a packet loss rate of .001, sending 4000 messages using 9 bits for the header (1 bit for the sequence number and 8 bits for the mode and power fields). The x-axis shows the partitioning of bits between the power field and the mode field. The y-axis shows messages/packet, hence higher values represent more efficient performance. For this particular loss rate, header size and number of message being sent, optimal performance is achieved by using 7 bits for the mode field and 1 bit for the power field.

References

- [AAF⁺91] Y. Afek, H. Attiya, A. Fekete, M. J. Fischer, N. Lynch, Y. Mansour, D-W. Wang, and L. D. Zuck. Reliable communication over unreliable channels. Technical Report YALEU/DCS/TR-853, Yale University, 1991.
- [AFWZ89] Hagit Attiya, Michael J. Fischer, Da-Wei Wang, and Lenore D. Zuck. Reliable communication using unreliable channels. Manuscript, May 1989.
- [BSW69] K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson. A note on reliable full-duplex transmission over half-duplex links. *Communications of the ACM*, 12:260–261, 1969.
- [GK82] D. Greene and D. Knuth. *Mathematics for Analysis of Algorithms*. Birkhäuser, Boston, M.A., second edition, 1982.
- [LLT92] R.E. Ladner, A. LaMarca, and E. Tempero. Counting protocols for reliable end-to-end transmission. Technical Report 92-10-06, University of Washington, Department of Computer Science and Engineering, 1992.

- [LMF88] N. Lynch, Y. Mansour, and A. Fekete. The data link layer: Two impossibility results. In *Seventh Annual ACM Symposium on Principles of Distributed Computing*, August 1988.
- [MS89] Y. Mansour and B. Schieber. The intractability of bounded protocols for non-FIFO channels. In *Eighth Annual ACM Symposium on Principles of Distributed Computing*, August 1989.
- [Pos81a] J. Postel. Internet protocol. Network Working Group Request for Comments 791, September 1981.
- [Pos81b] J. Postel. Transmission control protocol. Network Working Group Request for Comments 793, September 1981.
- [Sta91] W. Stallings. *Data and Computer Communications*. Macmillan Publishing, New York, N.Y., third edition, 1991.
- [Tan88] A. S. Tanenbaum. *Computer Networks*. Prentice-Hall, Inc., Englewood Cliffs, N.J., second edition, 1988.
- [Tem90] E. Tempero. Network protocols for non-FIFO channels. Ph.D. Dissertation, University of Washington. Technical Report No. 90-09-03, Department of Computer Science and Engineering, September 1990.
- [TL90] E. Tempero and R. Ladner. Tight bounds for weakly bounded protocols. In *Ninth Annual ACM Symposium on Principles of Distributed Computing*, pages 205–218, August 1990.
- [WZ89] D-W. Wang and L. D. Zuck. Tight bounds for the sequence transmission problem. In *Eighth Annual ACM Symposium on Principles of Distributed Computing*, August 1989.

Figures

```
SENDER  
  
BeginSend  
   $n, u_S \leftarrow 0$   
  REPEAT  
    inc( $n$ )  
     $X \leftarrow \text{read}$   
     $[d_S] \leftarrow \text{SendMsg}(n)$   
     $u_S \leftarrow u_S + d_S$   
  END REPEAT  
  
SendMsg( $n$ ) RETURNS  $[d_S]$   
   $Req, d_S \leftarrow 0$   
  IF  $n \neq 1$  THEN  
    send( $n \bmod 2, X$ )  
  ENDIF  
  REPEAT  
     $pkt \leftarrow \text{receive}$   
    CASE  
      ◦  $pkt = (n \bmod 2, \text{"request"})$   
        send( $n \bmod 2, X$ )  
      ◦  $pkt = (n \bmod 2, \text{"restart"})$   
        inc( $d_S$ )  
        send( $n \bmod 2, X$ )  
      ◦  $pkt = (n + 1 \bmod 2, \text{"request"})$   
        inc( $Req$ )  
        IF  $Req > u_S$  THEN  
          RETURN  
        ELSE  
          send( $n + 1 \bmod 2, \text{"null"}$ )  
        ENDIF  
      ◦  $pkt = (n + 1 \bmod 2, \text{"restart"})$   
        inc( $d_S$ )  
        send( $n \bmod 2, \text{"null"}$ )  
    END CASE  
  END REPEAT
```

Figure 1: The P_{one} protocol for the Sender

RECEIVER

BeginReceive

```
 $n, u_R \leftarrow 0$   
REPEAT  
   $\text{inc}(n)$   
   $[d_R] \leftarrow \text{ReceiveMsg}(n)$   
   $u_R \leftarrow u_R + d_R$   
   $\text{write}(Y)$   
END REPEAT
```

ReceiveMsg(n) RETURNS $[d_R]$

```
 $\forall x \text{ Req}[x] \leftarrow 0, d_R \leftarrow 0$   
 $\text{send}(n \bmod 2, \text{"request"})$   
REPEAT  
   $\text{pkt} \leftarrow \text{receive}$   
  CASE  
     $\circ \text{pkt} = (n \bmod 2, \text{"null"})$   
       $\text{send}(n \bmod 2, \text{"request"})$   
     $\circ \text{pkt} = (n \bmod 2, \neg \text{"null"})$   
       $Y \leftarrow \text{pkt.msg}$   
       $\text{inc}(\text{Req}[Y])$   
      IF  $\text{Req}[Y] > u_R$  THEN  
        RETURN  
      ELSE  
         $\text{send}(n \bmod 2, \text{"request"})$   
      ENDIF  
     $\circ \text{pkt} = \text{timeout}$   
       $\text{inc}(d_R)$   
       $\text{send}(n \bmod 2, \text{"restart"})$   
  END CASE  
END REPEAT
```

Figure 2: The P_{one} protocol for the Receiver

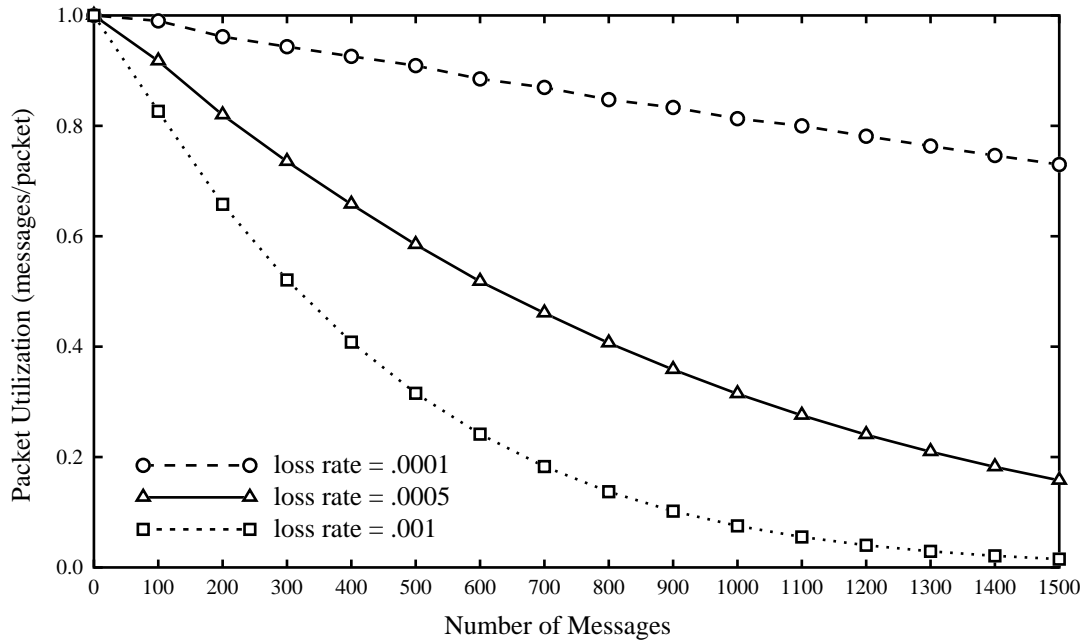


Figure 3: Performance of P_{one} in a statistical FIFO channel for various loss rates

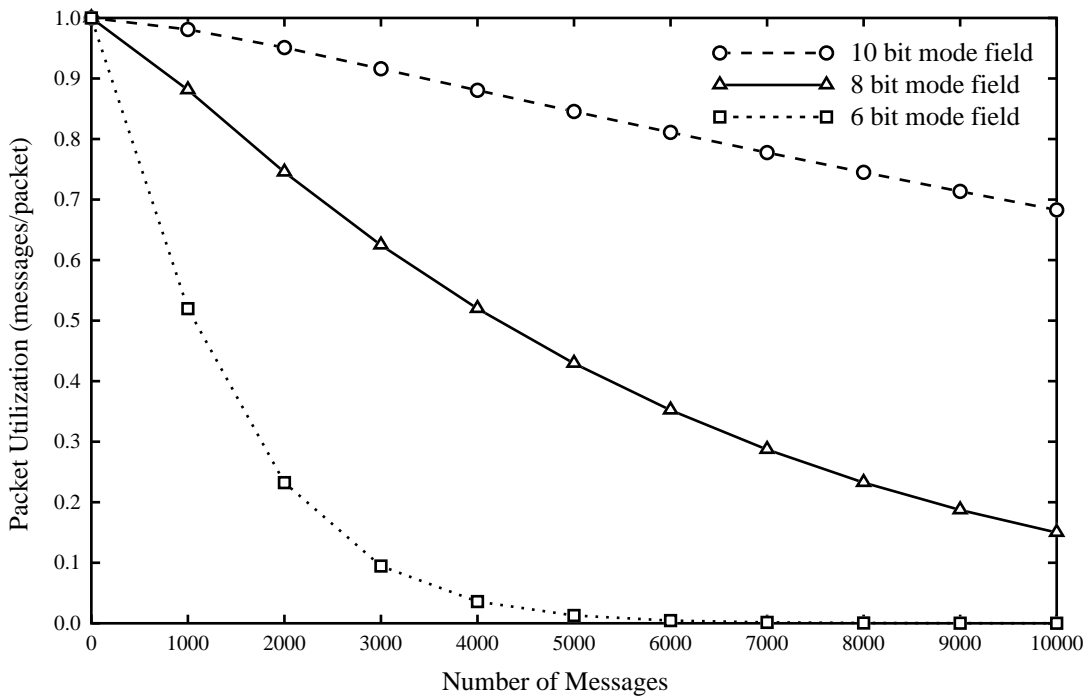


Figure 4: Performance of P_{mode} in a statistical FIFO channel with loss rate .05 and various size mode fields

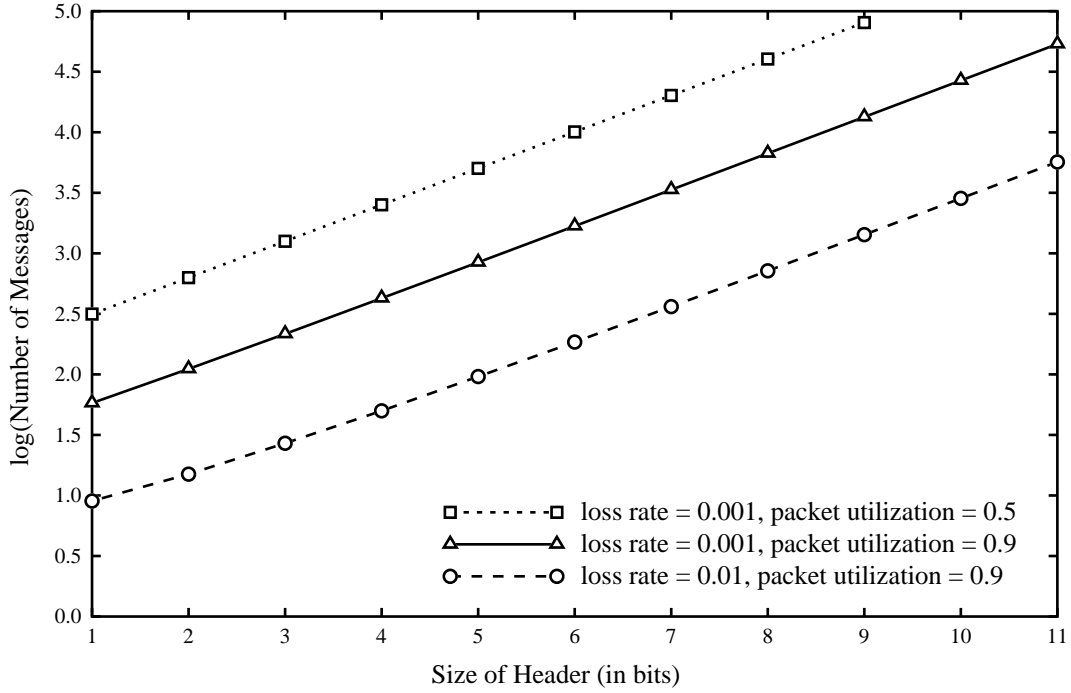


Figure 5: Header size vs. number of messages that can be sent using P_{mode} for various loss rates and packet utilizations

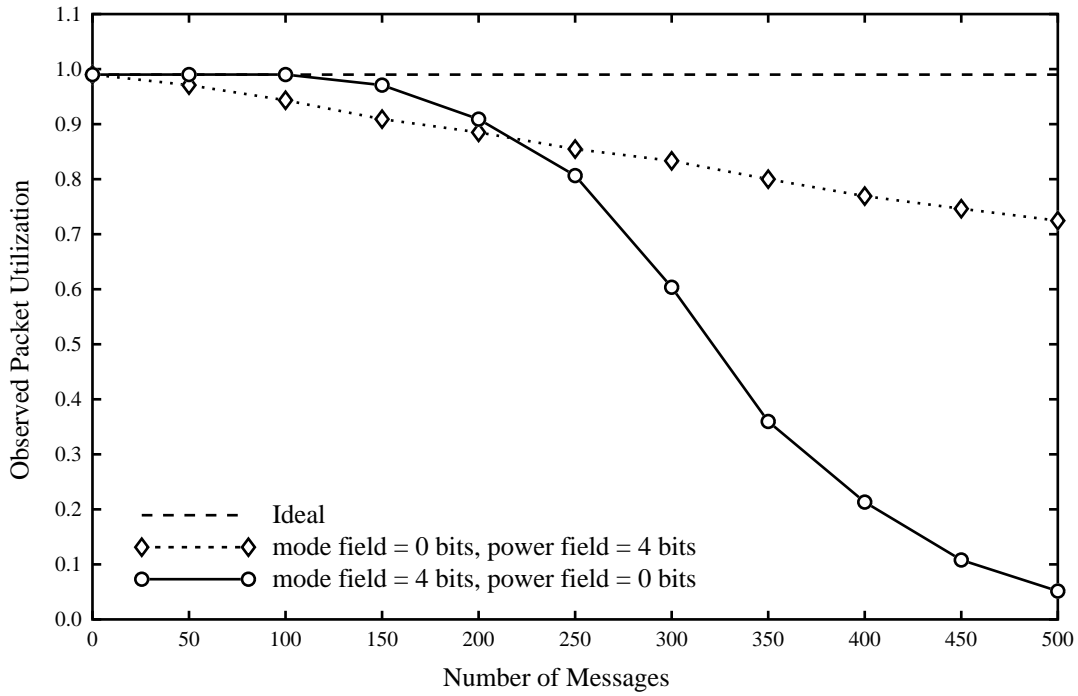


Figure 6: Comparison of the effectiveness of mode bits and power bits on packet utilization in P_{mp} . Simulations used a loss rate of .005 and data points represent the average of 100 simulated runs.

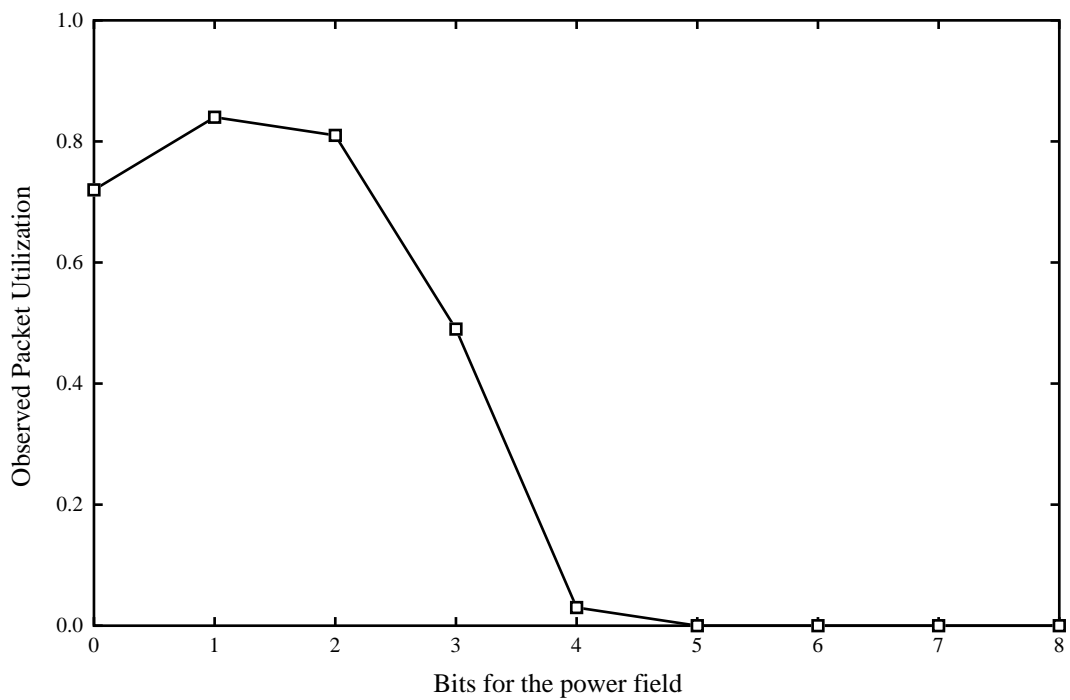


Figure 7: Packet utilization curve as a function of the partitioning of the header bits between the power and the mode field. Simulations sent a sequence of 4000 messages using a loss rate of .01, and a total of 8 bits for the mode and power fields. Data points represent the average of 100 simulated runs.