LIDS-P-2190

PERFORMANCE ANALYSIS OF POLL-BASED RETRANSMISSION SCHEMES

August, 1993

Jane M. Simmons
MIT, Laboratory for Information and Decision Systems

Abstract: Many connections require that the source retransmit any data message not correctly delivered to the destination. This paper focuses on schemes that rely on a polling mechanism to trigger the retranmissions. A delay criterion is presented that can be used to gain insight into the performance of general retransmission schemes; this criterion is then used to evaluate poll-based schemes. It is shown that the biggest drawback of poll-based schemes is that they potentially can result in large retransmission delay when bursts of messages are lost. Their main advantage is that if data arrives in-sequence at the receiver, then unnecessary retransmissions can be eliminated. Thus, poll-based schemes are best suited for systems where bandwidth efficiency is of greater concern than low delay.

This work was funded by the NSF, DARPA, and ATT Bell Laboratories.

1. INTRODUCTION

Many connections require that the source retransmit any data message not correctly delivered to the destination. There are a variety of methods that can be used to indicate to the source that a message needs to be retransmitted. The methods differ in the delay until the source performs the retransmission, the amount of overhead and complexity involved, and whether unnecessary retransmissions can occur. We focus our analysis on retransmission schemes that rely on a polling mechanism.

The polling mechanism can be either explicit or implicit. For example, the source can send an explicit poll message indicating which messages it has sent and requesting that the receiver send a status message indicating which messages it has received. Or, a data message can serve as an implicit poll. For example, assume we have a system where messages are expected to arrive in sequence. Then the arrival of a message with sequence number SN can be considered to be a poll questioning the arrival of all messages with sequence number less than SN.

In Section 2, we present a delay criterion that can be used to gain insight into the performance of general retransmission schemes; in Section 3, we use this criterion to evaluate poll-based schemes. We show that the biggest drawback of poll-based schemes is that they potentially can result in large retransmission delay when bursts of messages are lost. Their main advantage is that if data arrives in-sequence at the receiver, then unnecessary retransmissions can be eliminated.

One alternative to polling is to rely on a timer to trigger retransmissions. In a typical timer implementation, if an acknowledgement of a message is not received by a certain time, the source retransmits the message. Timer-based schemes potentially lead to many unnecessary retransmissions. However, in certain systems, they result in low retransmission delay. We review timer-based schemes in Section 4 (also see [1,2]), and compare them with poll-based schemes in Section 5. Our analysis shows that poll-based schemes are best suited for systems where bandwidth efficiency is important. In the past bandwidth efficiency was very important due to the high cost of data transmission; in the future, low delay will likely be more important. Timer-based schemes are capable of producing low retransmission delay if the round trip delay is not highly varying; they perform poorly when it is highly varying, as is often the case for low speed systems. However, in practice, systems designed for low speed traffic, such as the Transport Control Protocol (TCP) and ISO TP4, employ a timer scheme[1,3,4], and high speed systems currently under development, such as the Asynchronous Transfer Mode (ATM) and Xpress Transport Protocol (XTP), include a poll-based scheme [5,6,7].

2. RELATIVE EXCESS DELAY

In this section, we examine the various components of retransmission delay for a given point-to-point connection. We assume that retransmissions are performed on an end-to-end basis and that the unit of retransmission is a data message. Throughout this section we make the following simplifying assumptions: the data rate of the connection is constant, the source always has data to send, messages arrive at the destination in the same order in which they were sent by the source, packets are fixed size, data messages in the connection are comprised of N packets, and control messages are comprised of one packet.

We consider the round trip delay (RTD) of the connection to be the total propagation delay, queueing delay, and processing delay from source to destination and back. In high speed systems such as ATM, the RTD is expected to be almost constant. However, in low speed systems such as TCP/IP, the RTD may be highly variable. (This is discussed further in Section 4.) However, for our initial analysis, we will assume the RTD is fixed and that the delay from source to receiver is the same as from receiver to source.

Let R be the data rate of the connection, in terms of the number of packets transmitted per RTD. Throughout our analysis, the unit of time is the amount of time it takes to transmit one packet. Thus, the RTD is R time units. As a point of reference, we refer to networks with an R of about 1 or smaller as low speed networks; networks with an R on the order of 100 or higher can be considered to be high speed networks.

Consider any data message and assume the transmission of the message starts at time $\mathcal{T}$. Since we assume the transmission rate and RTD are fixed, the complete message is not expected to arrive at the destination until time $\mathcal{T}$+ N + .5R. N is the transmission delay in sending the N packets of the message, and .5R is the delay from source to destination. Thus, N + .5R represents the delay if the original transmission of the message is successful.

In order for the source to learn whether a transmitted message has been received correctly, it must allow time for the receiver to send a control message. For example, the receiver can send a positive acknowledgment (ACK) if a message is received successfully, or it can send a negative acknowledgement (NACK) if a message needs to be retransmitted. We assume that the receiver cannot make a decision as to whether or not a message needs to be retransmitted until time $\mathcal{T}$+ N + .5R. (Since data is assumed to travel at a fixed rate, a lost message can actually be detected earlier than time $\mathcal{T}$+ N + .5R; for simplicity, we assume that the receiver must always wait until this time before deciding the message needs to be retransmitted.) Thus, the earliest time at which the source can learn the status of a message is $\mathcal{T}$+ (N + .5R) + (1 + .5R). The '1' term is the transmission delay in sending a control message, and .5R is the delay from destination to source. Assuming the source waits until $\mathcal{T}$+ N + R +1 before

retransmitting a message, then the earliest time the retransmitted message could arrive at the destination is $\mathcal{T}$ + 2N + 1.5R + 1. Thus, the minimum delay due to retransmission is: ($\mathcal{T}$ + 2N + 1.5R + 1) - ($\mathcal{T}$ + N + .5R) = N + R + 1. We refer to this as the minimum retransmission delay. (In Section 3.2, we will consider a case where the minimum retransmission delay is forced to be larger than N + R + 1.)

The minimum retransmission delay can potentially be achieved by a timer-based scheme, although this would require that the source know the RTD precisely. For example, if an ACK of a message is not received N + R + 1 time units after the message is sent, then the message is retransmitted. However, not all retransmission schemes can attain the minimum retransmission delay. For example, there may be a delay from the time the destination determines the status of the message until the time it actually sends a control message back to the source indicating the status. Or, the source may not immediately retransmit a message once it learns that a retransmission is necessary. For example, if two messages are NACKed in one control message, then one of the retransmissions will have to be delayed while the other is performed. Thus, the actual retransmission delay may be N + R + 1 + $\mathcal{E}$, where we refer to $\mathcal{E}$ as the excess retransmission delay. As we will see in the sections below, the excess retransmission delay greatly depends on our choice of retransmission scheme.

The actual value of $\mathcal{E}$ is not the crucial factor to consider, but rather the relative value of $\mathcal{E}$ compared to the minimum retransmission delay. We define the relative excess delay $\mathcal{D}$ to be:

$$\mathcal{D} = \frac{\mathcal{E}}{N + R + 1} \quad (1)$$

Before proceeding, we discuss why the relative excess delay is important. First, consider a high speed network. High speed networks such as ATM are likely to use a selective repeat scheme, thus necessitating a resequencing buffer at the destination. (For background information on selective repeat and Go Back N systems, refer to [1,8,9,10].) The retransmission delay affects how large the buffer should be. The relative excess delay $\mathcal{D}$ affects the relative extra buffer space that is needed. Consider a simplistic example where if $\mathcal{D}$ were 0, the buffer space would need to be 10 Mb. and if $\mathcal{D}$ were 10%, the buffer space would need to be 11Mb. Even though the size of the increment, 1Mb., is large, there is not that much difference in requiring a 11Mb. buffer over a 10Mb. buffer. From equation (1), we see that as the data rate increases, $\mathcal{D}$ tends toward $\frac{\mathcal{E}}{R}$. Thus, at very high speeds, we are essentially looking at the excess delay relative to the round trip delay.

Next, consider a low speed network. Typically, on low speed networks, Go Back N is used rather than selective repeat. One of the chief concerns on a low speed network is the

amount of bandwidth that is used. If a data message is lost and Go Back N is used, then the minimum amount of wasted bandwidth will be N + R + 1 packet transmissions. $\mathcal{D}$ represents the relative additional amount of wasted bandwidth. We see from equation (1) that if R is very small compared to N, then $\mathcal{D}$ is close to $\frac{\mathcal{E}}{N}$. Thus, at very low speeds, we are essentially looking at the excess delay compared to the transmission delay of one message.

By looking at the relative excess delay, we favor schemes where $\mathcal{E}$ is small compared to the minimum retransmission delay. This may not be the most important criterion, however, for connections carrying delay sensitive traffic. For these connections, overall delay is the primary concern; how the delay is apportioned between the minimum retransmission delay and the excess delay is of secondary importance. For example, a scheme where the excess delay and the minimum retransmission delay are equal and small in magnitude may be appropriate for delay sensitive traffic. However, it would not appear to be a good scheme using the relative excess delay criterion. Nevertheless, for many connections, the relative excess delay criterion is a meaningful measure of performance. In the next section, we will use this criterion to gain insight into the performance of poll-based retransmission schemes.

3. PERFORMANCE OF POLL-BASED RETRANSMISSION SCHEMES

In the basic polling scheme, the source periodically sends poll messages to the receiver indicating which data messages have been sent. The receiver responds to the poll by transmitting a status message indicating which of these data messages have not been received. The source uses the status message to determine which data messages can be released from the retransmission buffer and which data messages need to be retransmitted. We start off by analyzing this simple scheme; at the end of the section we consider adding various options.

3.1 Random Losses

First consider the case where polls and status messages are not lost, and no more than one data message is lost in between poll transmissions (i.e., a status message never generates more than one retransmission). We assume that the receiver sends a status message immediately after receiving a poll, and any necessary retransmissions are performed as soon as the status message is received. With these assumptions, the excess retransmission delay $\mathcal{E}$ for any errored message consists of the delay from when the message should have arrived at the destination until the time a status message is sent NACKing the message. As shown if Figure 1, $\mathcal{E}$ depends on how soon a poll is sent after the errored data message.

Let I equal the number of data messages that are sent in between poll transmissions. The I messages can be original transmissions or retransmissions. The expected value of $\mathbb{E}$ for the simple polling scheme, using the assumptions above, is:

$$E = \text{Expected Value of } \mathbb{E} = \frac{(I-1)N}{2} + 1 \qquad (2)$$

The '1' term represents the transmission delay of sending the poll message. We combine equations (1) and (2) to derive the expected relative excess delay for the polling scheme:

$$D = \text{Expected Value of } \mathbb{D} = \frac{(I-1)N + 2}{2N + 2R + 2} \qquad (3)$$

In practice, it makes sense to use only integral values of I. A poll sent in the middle of a data message generates the same status information as a poll sent at the beginning of the data message, but generates it at a later time. Thus, only those values of D that correspond to integral values of I can be attained. The minimum possible relative excess delay for a polling scheme is attained when I equals 1. This corresponds to sending a poll after every message.

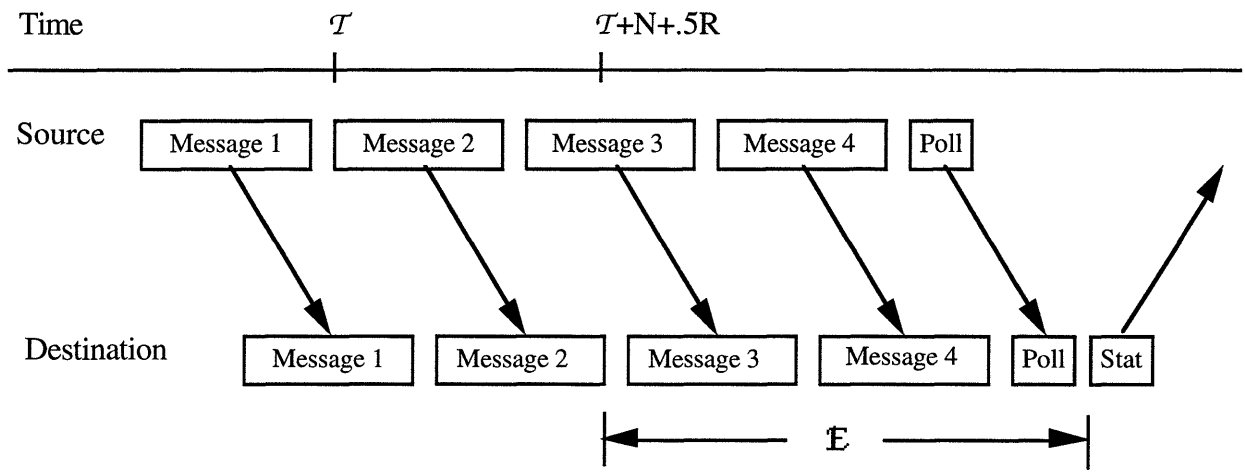


Figure 1 $\mathbb{E}$ represents the delay from the time message 2 should have arrived successfully at the destination to the time the destination begins to transmit a status message NACKing message 2.

3.1.1 Polling Frequency

Assume a data connection requires that the expected relative excess delay be no more than some D_{max} (for the case when random messages are lost). Rearranging equation (3), the source can calculate the appropriate value for I by:

$$I = \left\lfloor \frac{D_{max}(2N + 2R + 2) + N - 2}{N} \right\rfloor \qquad (4)$$

(where $\lfloor \bullet \rfloor$ indicates the integer portion)

Some of the polling scheme proposals refer to sending a certain number of polls per RTD rather than sending a poll after every I data messages. Given the target value D_{max}, and letting the number of polls that should be sent per RTD be F, then:

$$F = \frac{R}{N\left\lfloor \frac{D_{max}(2N + 2R + 2) + N - 2}{N} \right\rfloor + 1} \qquad (5)$$

As R increases for a fixed N, we see that I increases toward $\frac{2D_{max} R}{N}$ and F increases toward $\frac{1}{2D_{max}}$. Thus, more data messages are sent in between poll transmissions but more polls are sent per RTD. Also, the polling rate in terms of polls per RTD becomes independent of N.

3.1.2 Polling Overhead

From the standpoint of delay, polls should be sent frequently (i.e., the smaller I is, the smaller D is in equation (3)). However, if explicit poll messages are sent, then a higher polling frequency leads to greater overhead. A poll is comprised of one packet and one poll is sent after every NI data packets. Letting V equal the fraction of traffic from source to destination that is dedicated to polling, we see that:

$$V = \frac{1}{NI + 1} \qquad (6)$$

(We ignore the overhead in the reverse direction, but for each poll arriving at the destination there is a status message sent back to the source.)

To see the tradeoff between expected relative excess delay and overhead as a function of N and R, we combine equations (3) and (6) to get:

$$V = \frac{1}{D(2N + 2R + 2) + N - 1} \qquad (7)$$

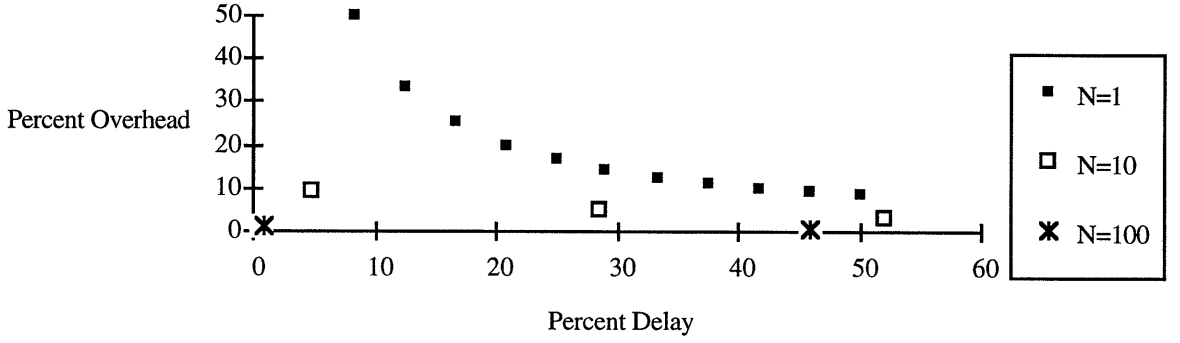
This tradeoff is plotted in Graph 1 for R equal to 10 and in Graph 2 for R equal to 1000. (A data rate of 100 Kb/sec, an RTD of 50 msec., and a packet size of 500 bits correspond to an R of 10.) As stated above, only certain values of D can be attained due to the integer constraint on I. Thus, for R equal to 10 only the discrete points that correspond to integral values of I are plotted. For R equal to 1000, the discrete points were close enough together that continuous curves are plotted.

If N is large, then both low expected relative excess delay and low overhead can be achieved (although if R is small, it can only be achieved if a poll is sent after each data message). If R and N are small, then there is a greater tradeoff between delay and overhead. For example, for R equal to 10 and N equal to 1, an expected relative excess delay of 10% can

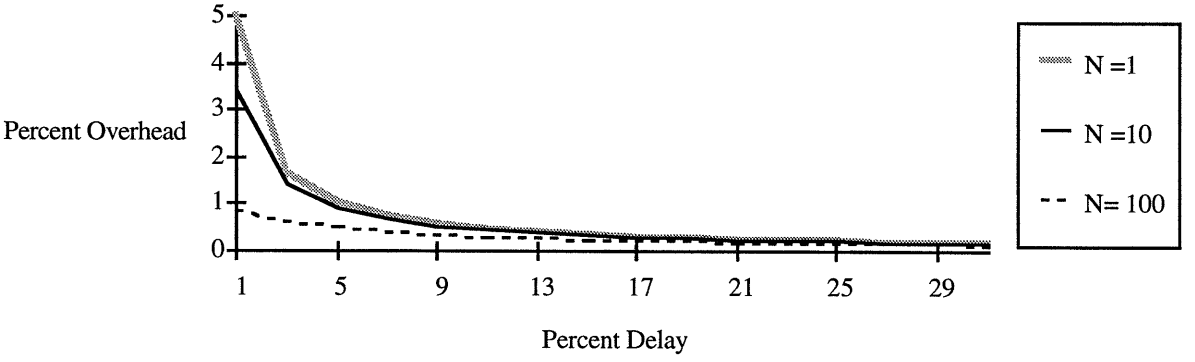
be attained, but only at the expense of about 50% overhead. As R gets larger, the overhead V tends towards $\frac{1}{2RD}$ and is thus independent of N. This is illustrated on Graph 2. For R equal to 1000, we can attain a 5% expected relative excess delay with less than 1% overhead, regardless of the size of the message.

Note that the polling overhead may not be a critical factor. As we discuss in Section 3.5.2, it may be possible to piggyback the polls on data messages, so that the overhead is reduced to almost zero.

Graph 1 Percent Overhead vs. Percent Expected Relative Excess Delay R=10



Graph 2 Percent Overhead vs. Percent Expected Relative Excess Delay R=1000



3.1.3 Worst Case vs. Average Case

Above we considered the average excess delay; we should also consider the worst case excess delay. Let E equal the expected excess delay as defined in equation (2). Assuming polls and status messages are not lost, then the worst case scenario in terms of delay is when the

message transmitted immediately after a poll is lost. In this case, the delay is approximately double that of the average case. This is shown in Figure 2.

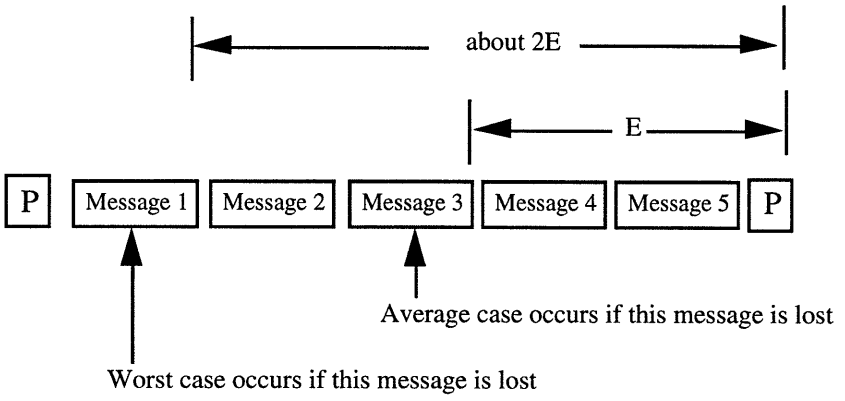


Figure 2 Average and worst case delay if poll and status message are not lost.

3.2 Burst Losses

In Section 3.1, we assumed that a status message never generated more than one retransmission. Thus, when the status message is received at the source, any NACKed message can be immediately retransmitted. In this section, we consider burst losses, where many consecutive messages are lost. First, consider the case where message X and message X+1 are both NACKed in the same status message. Message X+1 will not be retransmitted immediately; it will be delayed by an extra N time units while message X is retransmitted before it.

Next consider the case where all I data messages that are sent in between poll transmissions are dropped. Refer to Figure 3, where we assume I equals 3. Assume the poll that follows the dropped messages is received correctly, and the corresponding status message is received at the source. This status message will NACK all I messages. From the figure, we see that for all I messages, there will be a delay of NI + R + 2 time units from the time the message is originally transmitted until the time it is retransmitted. Since the minimum possible retransmission delay is N + R + 1, each message suffers an excess delay of (I-1)N + 1. Above, where we assumed only one of the I messages is lost, the expected excess delay is $\frac{(I-1)N}{2} + 1$.

If the burst loss is due to a burst error along the data path, then it is likely the poll messages will be lost also. Or, if the burst loss is due to congestion, and polls do not have a higher priority than data, then it is likely that polls will be dropped. If no polls or data messages arrive at the receiver, then no status messages will be sent. Since the source cannot determine which

messages have been lost until a status message arrives, the lack of status messages will delay the retransmission process.

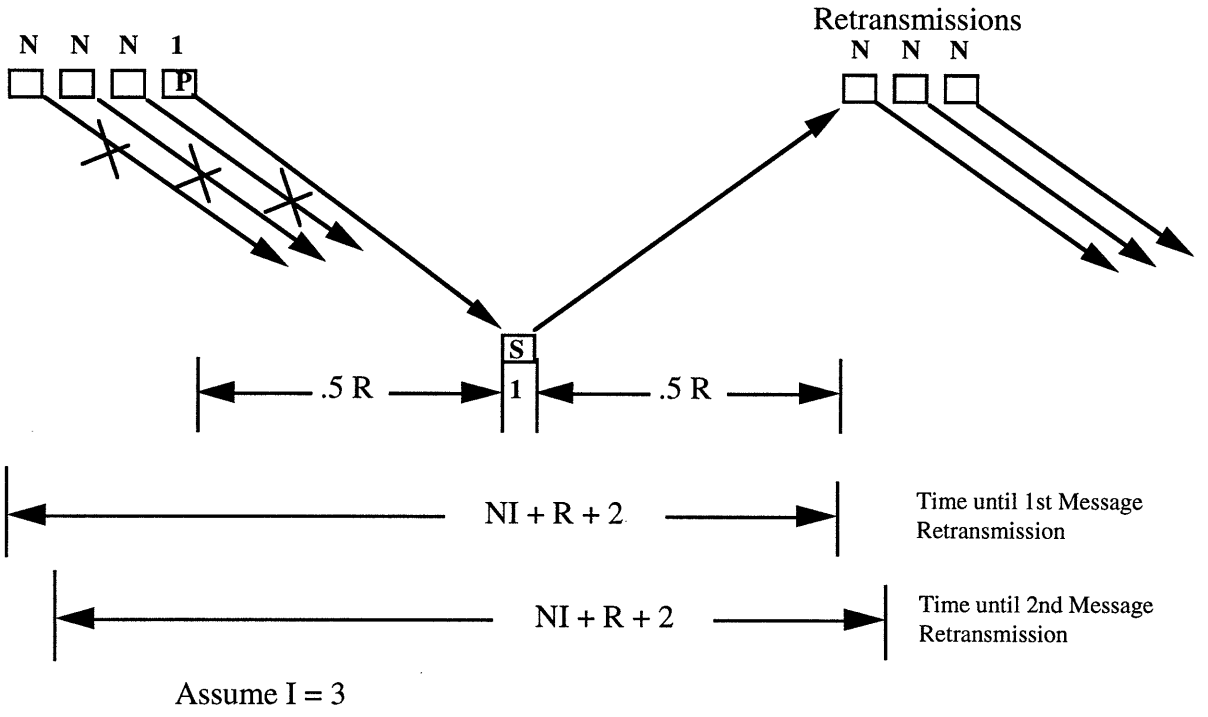


Figure 3 Delay if all I data messages are lost.

Assume a burst loss starts at time $\mathcal{T}$ and ends at time $\mathcal{T}$+ B, where B is less than R + N + 1, i.e., all transmissions from time $\mathcal{T}$ to $\mathcal{T}$+ B are dropped. Define a successful poll as one that arrives successfully at the receiver and whose corresponding status messages arrives successfully at the source. Assume the first successful poll after the burst ends is sent at time $\mathcal{T}$+ B + δ, as shown in Figure 4. Then the first retransmission will occur at time $\mathcal{T}$+ B + δ + R + 2 (the '2' term is due to the transmission time of the poll and of the status message). If the excess delay had been 0, the first retransmission would have been sent at time $\mathcal{T}$+ R + N + 1. Thus, in this scenario, the polling scheme results in an excess delay of B + δ + 1 - N. The relative excess delay is:

$$\frac{B + \delta + 1 - N}{R + N + 1} \qquad (8)$$

If the first poll sent after the burst ends is successful, then δ lies between 0 and NI.

We see that the excess delay in this scenario has two main components: the delay until the burst error ends (i.e., B) and the delay until a poll is sent (i.e., δ). In Section 3.1, where we discussed the scenario of just a single message being dropped (as opposed to a burst of

messages being lost), the excess delay was composed of the delay until a poll is sent. Thus, when burst errors occur, and the length of the burst error is shorter than R + N + 1, the excess delay essentially increases by the length of the burst error. (The actual amount of the increase in excess delay will depend on where in the polling 'cycle' the burst error starts and ends.)

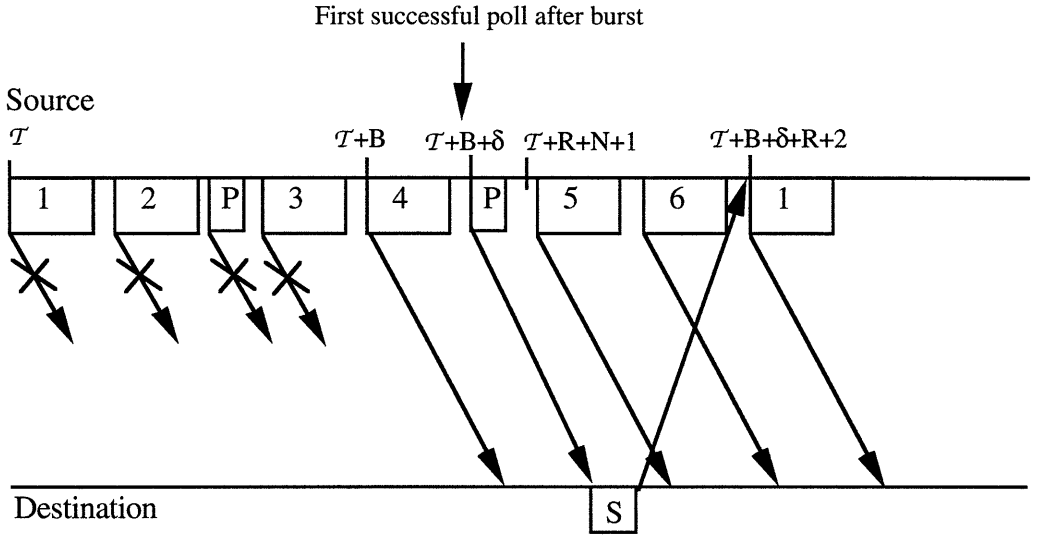


Figure 4 Ideally, message 1 would be retransmitted at time $\mathcal{T}$+ R + N + 1. However, message 1 will not be retransmitted until a poll sent after it reaches the destination successfully and triggers a successful status message. Thus, the retransmission mechanism is susceptible to error in either direction.

Next, consider the case where B is greater than R + N + 1. It would not be desirable to retransmit a message at time $\mathcal{T}$+ R + N + 1, since it would be lost again as part of the burst loss. Ideally the first retransmission would be sent at time $\mathcal{T}$+ B. Refer to Figure 5. Assume that the first successful poll after the burst ends is sent at time $\mathcal{T}$+ B + δ. The first retransmission will occur at time $\mathcal{T}$+ B + δ + R + 2 rather than $\mathcal{T}$+ B. Thus, in this scenario, the excess delay is δ + R + 2. The relative excess delay, according to the modified definition, is:

$$\frac{\delta + R + 2}{B} \qquad (9)$$

If the first poll sent after the burst ends is successful, then δ lies between 0 and NI.

The excess delay in this scenario has two main components: the delay until a poll is sent (i.e., δ) and the round trip delay until the corresponding status message is received (i.e., R). In all of the other scenarios discussed thus far, the delay in waiting for the status message to arrive was part of the minimum retransmission delay and not the excess delay (i.e., in other scenarios the source needed to wait a minimum of one round trip delay in order to allow feedback from

the destination time to arrive; in this scenario, we assume that the duration of the burst error provides enough time for any feedback to reach the source so that ideally the source would start to retransmit as soon as the burst error ends.) Thus, compared to the scenario where single messages are lost, the excess delay for the long burst error scenario essentially increases by the amount of one round trip delay. (The actual amount of the increase in excess delay will depend on where in the polling 'cycle' the burst error starts and ends.)

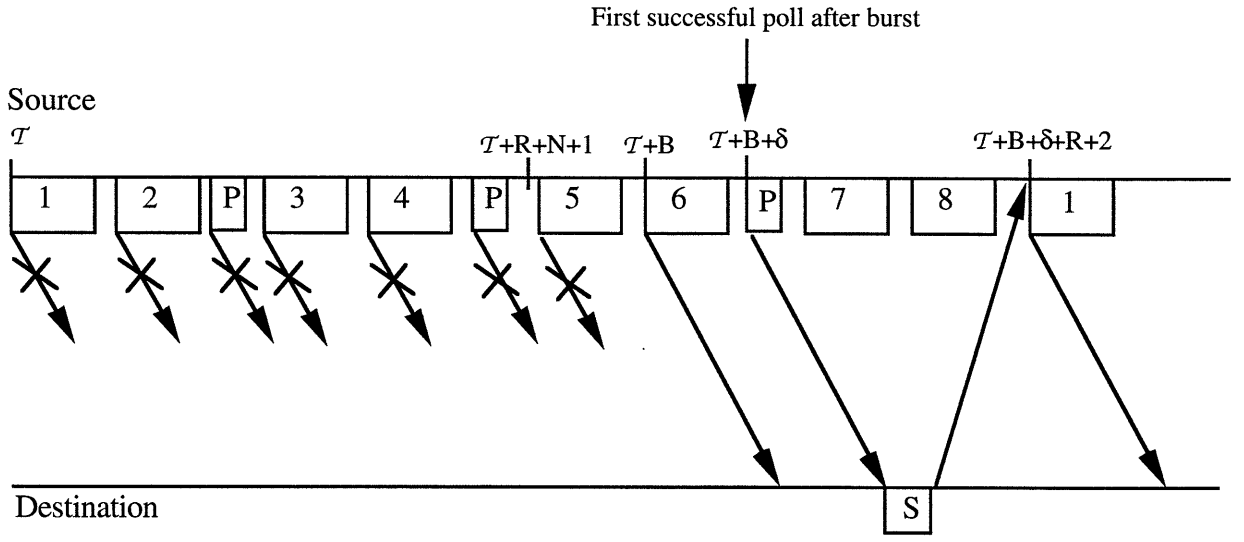


Figure 5 It is not desirable to retransmit message 1 at time $\mathcal{T}$+ R + N + 1, since it would be dropped again. Ideally, it would be retransmitted at time $\mathcal{T}$+ B. However, the retransmission will not occur until a poll reaches the destination and its corresponding status message reaches the source.

As we can see from these examples, burst losses result in an increase in the excess delay. This delay increase is desirable in the following situation: assume the burst loss is due to congestion and assume window-based flow control is used. Window-based flow control does not exercise control over retransmissions (unless the window shrinks sufficiently); the source dumps a retransmission into the network whenever a NACK arrives. If status messages are not being sent by the receiver, then NACKs will not be received by the source. Thus, poll-based schemes provide more time for the congestion to dissipate.

If rate-based flow control is used, however, the increase in excess delay is not desirable. In rate-based flow control, the source sends α messages per second, for some α, whether the messages are original transmissions or retransmissions. Thus, the lack of status messages will not decrease the traffic being sent into the network. It just means that until status messages arrive NACKing the lost data, the data messages that are sent by the source will be original transmissions rather than the necessary retransmissions. Of course, it is possible the source

will decrease its data rate to β messages per second after it has not received a status message for a long time. But given that it is sending β messages per second, it would be preferable from the standpoint of delay that these be the necessary retransmissions rather than new transmissions (since the receiver must deliver the messages in proper sequence). Thus, the fact that status messages are not being sent by the receiver is undesirable in this scenario. (However, note that a rate-based scheme would likely be implemented with some sort of restriction on the transmission window so that the lack of NACKs in the polling scheme would eventually lead to the end of the send window being reached; at this point the extra delay in the polling scheme would be beneficial if congestion is still present in the network.) If the burst loss is due to a burst error along the data path, then the increase in excess delay is undesirable.

3.3 Prevention of Unnecessary Retransmissions

One advantage of using a polling scheme is that unnecessary retransmissions can be prevented if messages arrive at the receiver in the same order in which they are transmitted. The general idea is that if a poll is sent immediately after message number X, then by the time the poll arrives at the destination, all messages with sequence number less than or equal to X should have arrived already. Thus, the receiver knows that if any of these messages have not arrived, they need to be retransmitted. It is assumed the poll contains some indication that all messages up to sequence number X have been transmitted.

Of course, a second poll could arrive at the destination before the retransmissions have been received. This would cause a second status message to NACK the lost messages, which could lead to unnecessary retransmissions. To avoid this, there must be some way of associating a NACK with the poll that generated it, and there must be some way of determining whether a retransmission has been sent before or after the poll. ATM makes use of poll sequence numbers to accomplish this; refer to [5] for the full details of the scheme. In [11], we formally prove that the method used in ATM does prevent unnecessary retransmissions, under certain reasonable conditions.

3.4 RTD Estimate

From equation (5), we see that in order to determine the frequency of polls, the source needs to know the value of R. R is defined as the number of packets transmitted in one RTD. Above, we assumed that the RTD remained constant throughout the connection. In reality, the RTD will change due to varying queueing delays along the data path. Thus, the source needs to

monitor the delay between sending a poll and receiving an associated status message in order to estimate the RTD.

If the estimate of the RTD is too high, the excess delay will be higher than intended. If the estimate is too low, the overhead will be higher than necessary. However, even if the source incorrectly estimates the RTD, unnecessary retransmissions do not occur (assuming some sort of sequencing scheme is used). As we will see in Section 4, this is not the case for timer-based schemes.

3.5 Variations of the Simple Polling Scheme

3.5.1 Unsolicited Status Messages

One feature that can be added to a polling scheme is that in addition to sending status messages in response to polls, the receiver is permitted to send a status message whenever it determines that a data message has been lost. For example, consider a connection where data is expected to travel in order. If message number 2 arrives and message number 1 has not arrived, the receiver can assume message number 1 has been lost and immediately send a status message requesting the retransmission of message number 1 rather than waiting for a poll to arrive. This feature is included as part of the ATM proposal.[5] We will adopt the terminology of ATM and refer to these additional status messages as unsolicited STATs. We will refer to status messages sent in response to polls as solicited STATs.

Let's analyze the delay benefit provided by unsolicited STATs more precisely. Consider any lost data message. Assume that the message following the lost data message arrives correctly, and assume the source is sending messages at a steady rate. The excess delay E depends on which data message has been lost (see Figure 6). If the lost data message is immediately followed by a poll, then the lost message is NACKed in a solicited STAT and E equals one (the unit of time is the time to transmit one packet). In all other cases, the lost data message will be initially NACKed in an unsolicited STAT. The unsolicited STAT is sent after the destination receives and reassembles the next data message; thus, E equals N.

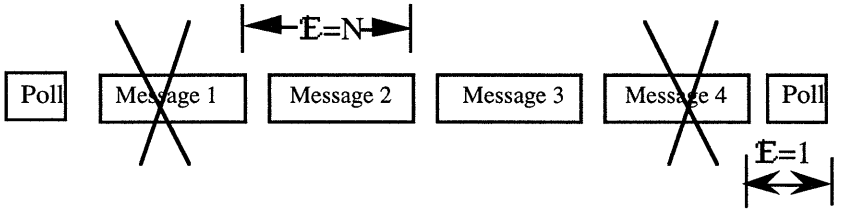


Figure 6 Excess retransmission delay when unsolicited STATs are used. If message 1 is lost, it will be NACKed in an unsolicited STAT; if message 4 is lost, it will be NACKed in a solicited STAT.

Assume polls are sent after every I data messages. The expected value of E is then:

$$E = \text{Expected Value of } E \text{ using unsolic. STATs} = \frac{(I-1)N}{I} + \frac{1}{I} \quad (10)$$

Combining equations (1) and (10), we get an expected relative excess delay of:

$$D = \frac{(I-1)N + 1}{I(N + R + 1)} \quad (11)$$

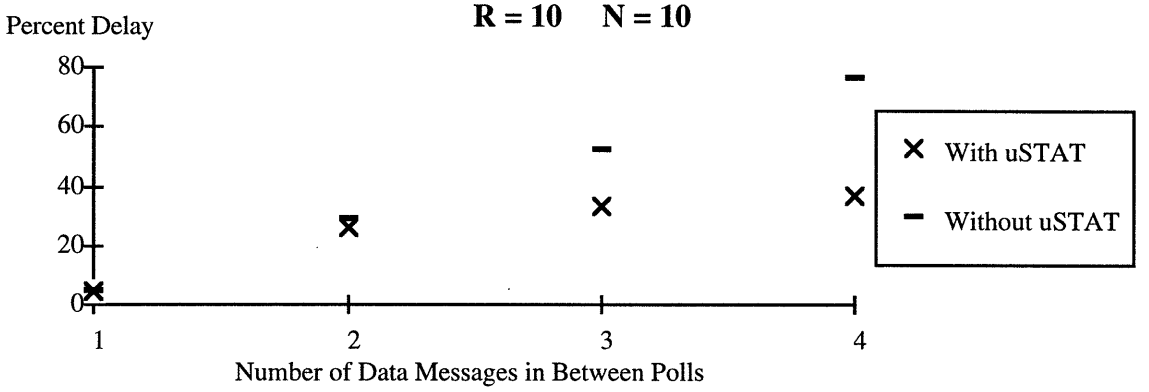
Comparing equations (2) and (10), we see that the delay benefit in using unsolicited STATs is greater the larger the value of I. This can be seen on Graphs 3 and 4. If I equals 1, a poll arrives after each data message; thus, the unsolicited STAT option would be unnecessary (unless a poll were lost). For some low and medium speed connections, I may need to be 1 in order to meet a certain delay goal. For example, assume it is desired that the expected relative excess delay be no more than 20%, and assume R is 10 and N is 10. From Graph 3, we see that I needs to be 1 in order to keep the expected relative excess delay below 20%. There would be no need to implement unsolicited STATs. At high speeds, even as I increases the relative excess delay remains small (see Graph 4); it is likely the polling mechanism would be implemented with I greater than 1. Thus, unsolicited STATs would provide a delay benefit for such connections. However, the reduction may be very small, e.g., on Graph 4, if I equals 6, the reduction in delay is less than 2%.

Note that equations (10) and (11) pertain to the case where random losses occur. If a burst of messages is lost, the excess delay will increase, as was described in Section 3.2. A retransmission will not occur until either a data or poll message gets through to the destination and its corresponding status message arrives at the source.

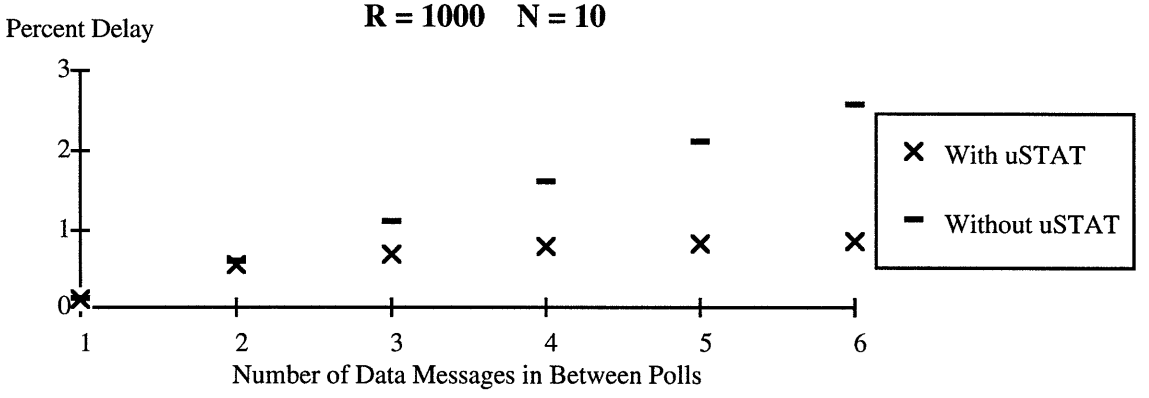
Unsolicited STATs can also be viewed as a means of reducing the polling overhead. For example, assume R is 10 and N is 10 and assume the desired maximum expected relative excess delay is 40%. Then, as shown on Graph 3, the interval between polls can be increased from 2 to 4 messages if unsolicited STATs are used. However, there are problems with reducing the polling rate too much. First, if an unsolicited STAT is lost en route to the source, there could be a long delay until the next STAT is sent. Second, if traffic is bursty and the last message in a burst is lost, it will not be NACKed until a poll arrives or data transmission resumes. (Actually, polls should probably be sent at the end of data bursts.) Third, status messages are also used to ACK data so that the source can release ACKed messages. Since unsolicited STATs are only sent when a message is lost, they cannot be relied upon for releasing data at the source. A lower polling rate would therefore necessitate larger buffers at the source.

Overall, we see that for certain connections, unsolicited STATs may provide a benefit in terms of delay and overhead, but the benefit may be insignificant. The drawback of using unsolicited STATs is the increased complexity of the polling scheme. The receiver must send two different types of status messages; these two types of status messages may need to be processed differently by the source (as is the case in ATM).

Graph 3 Percent Expected Relative Excess Delay vs. Interval Between Polls
R = 10 N = 10



Graph 4 Percent Expected Relative Excess Delay vs. Interval Between Polls
R = 1000 N = 10



3.5.2 Piggybacked Polls

In this section we consider a solicited STAT-only scheme where the polls are 'piggybacked' on data messages. It is assumed each data message contains a field that indicates whether the message also serves as a poll. The advantage of this option is that separate poll messages may not have to be sent, thus reducing the polling overhead.

Before discussing the drawbacks, we should comment on the implementation of piggybacked polls. The poll indicator field could be included in any packet of the data message. Nevertheless, we assume that the entire message must be received and reassembled at the

destination before the poll can generate a status message. From the point of view of delay, it would be better if the poll could generate a status message as soon as the packet that contains it arrives. However, this violates the principle of isolating layers within the network, since we assume retransmissions are handled only at the layer that deals with messages. Also, if a data redundancy check is included per-message rather than per-packet, then the integrity of the poll-carrying packet can only be checked once the whole message has been reassembled.

One drawback of piggybacking polls is that it may lead to greater delay. Consider the scenario where a data message is lost but the following message is received intact. If we are using a piggybacked poll scheme, and the following message is marked as a poll, then the excess retransmission delay is N time units, i.e., the time for the following data message to arrive. In a scheme where explicit one packet poll messages are sent after every data message, the excess delay is only one time unit.

In fact, the difference in delay may be even greater. The longer a message is, the greater the probability it will be lost; thus, a poll piggybacked onto a long data message is more likely to be dropped than a one packet explicit poll message. In general, we see that the larger N is, the greater the increase in expected excess delay when using piggybacked polls.

The third drawback of using piggybacked polls is that it may not be possible to totally rely on them as the sole means of polling. If there is no more data to be sent or if the data traffic is very bursty then explicit polls may need to be sent. This adds slightly to the complexity of the scheme, since the source has to monitor whether or not explicit poll messages are needed.

We conclude that using piggybacked polls makes the most sense when N is relatively small. With N small, the savings in overhead can potentially be large and the delay penalty relatively small.

3.5.3 Receiver Generated Status Messages

In the poll-based schemes discussed thus far, the source sends polls to the receiver and the receiver immediately responds to the poll by sending a status message. Another option is to have the receiver control when status messages are sent. Consider a scheme where the receiver relies solely on receiving a message with sequence number greater than X in order to determine that message X has been lost; neither the receiver nor the source maintains a timer for the purpose of generating retransmissions. This is similar to the unsolicited status message option discussed earlier except that the receiver controls when it sends the status message. As with the schemes discussed above, it is possible to eliminate unnecessary retransmissions in receiver-controlled schemes, assuming that data travels in sequence.

There are some potential problems with receiver controlled schemes if no retransmission timers are maintained. For example, if the last message of a burst is lost, the receiver will have no way of knowing the message was ever sent, and thus will not send a status message NACKing it until data transmission resumes. Another problem scenario is tied in with window-based flow control.[12] Assume that the window size is W; if the oldest unacknowledged message is X, then the source can't send messages past X+W-1. If W consecutive messages are lost, the protocol will deadlock.

There are several solutions to these problems, all of which add to the complexity of the protocol. The solution discussed in [12] is to send 'empty' data messages that are not subject to flow control restrictions. Empty messages could be sent if the source has no more data to send or if it has reached the end of its send window. Alternatively, the source could send a poll message rather than an empty data message. Also, the scheme could be modified so that a retransmission timer is kept at either the source or the destination. This would avoid the deadlock problems, although unnecessary retransmissions may occur.

The overhead should be less in receiver-based schemes. The receiver does not have to send a status message until it has something to NACK, or until it needs to free up space in the retransmission buffer at the source. Thus, fewer status messages would be sent in a system where the message loss rate is not high. Also, explicit poll messages do not have to be sent, except for the situations enumerated above. The minimum delay before a lost message is NACKed is N time units, i.e., the time for the next message to arrive. Thus, as N gets larger, the excess delay in receiver-controlled schemes increases.

The receiver-controlled scheme provides greater flexibility by allowing the receiver to send status messages when it wants. Other than that, it is not significantly different from a polling scheme where piggybacked polls and unsolicited STATs are allowed.

4. TIMER-BASED RETRANSMISSION SCHEMES

In this section, we examine schemes where retransmissions are triggered by a timer rather than a poll. We assume that the retransmission timer is maintained at the source (alternatively the timer could be kept at the receiver as is done in NETBLT[2,13,14]). After a message is transmitted, we assume the time of transmission is stored along with the message in the retransmission buffer. The receiver sends an ACK after each successful message arrival. As ACKs arrive, the source continually updates its estimate of the RTD. If the estimate of the RTD added to the timestamp of the oldest unacknowledged message is less than the current time, then the message should be retransmitted.

4.1 Delay Analysis

Assume packets are fixed size, messages are comprised of N packets, and the data rate is fixed. First, consider the ideal case where the RTD is known precisely. Using the conventions of Section 2, if a message is initially transmitted at time $\mathcal{T}$, an ACK of the message should arrive at the source at time $\mathcal{T}$+ N + R + 1. If an ACK has not been received by this time, the source assumes the message did not reach the destination successfully and retransmits the message.

Thus, in this ideal case where R is known, the excess delay is 0. Realistically, R is not known precisely. Let R_T be the true RTD and let R_E be the estimate of the RTD. If $R_E > R_T$, then the excess delay equals $R_E - R_T$. However, if $R_E < R_T$, then the timer will expire before the ACK could possibly have reached the source; this may result in an unnecessary retransmission.

In contrast to poll-based schemes, the performance of the timer-based scheme heavily relies on how well the source can estimate the RTD. If the RTD is not highly variable then the source should have a good estimate of the RTD, so that there is little excess delay. If the RTD does vary a lot, the inability of the source to accurately estimate the RTD may lead to large excess delay, or a large number of unnecessary retransmissions. The variability of the RTD depends largely on the queueing delays along the data path. Queueing delays are caused by packets arriving at an intermediate node faster than the node can process them, or by many packets at a node contending for the same output line. In either case, the packets must wait in a buffer until the node can process them (or if the buffer is full, the packets will be dropped). The time spent waiting in a buffer constitutes queueing delay and increases the effective RTD. For a given packet size and buffer size, queueing delays are potentially longer the lower the data rate of the network (assuming flow control is equally effective in both cases). Thus, the variability of the RTD is expected to be larger with low speed networks than with high speed networks. Thus, we expect the performance of timer-based schemes to be worse on low-speed networks.

4.1.1 Burst Losses

In Section 3, it was shown that in poll-based schemes, excess delay increases when a burst of messages is lost. In timer-based schemes, however, retransmissions are not delayed when burst losses occur. Assume the first message lost in the burst is originally transmitted at time $\mathcal{T}$, and assume messages are lost for the next B time units. First, consider the case where B is less than R + N + 1. Assuming the source knows the RTD precisely, it will begin to

retransmit the lost messages at time $\mathcal{T}$ + R + N + 1; thus, the relative excess delay will be 0. Thus, timer-based schemes are relatively unaffected by burst errors shorter than one round trip delay. As shown in Section 3.2, if a poll-based scheme is used, the expected excess delay increases by approximately the length of the burst.

Next, consider the scenario where B satisfies: $K(R + N + 1) < B \leq (K+1)(R + N + 1)$ for some $K \geq 1$. The first lost message will be retransmitted at times $\mathcal{T}$ + i(R + N + 1) for $1 \leq i \leq (K+1)$. The first K retransmissions will be lost as part of the burst; we assume the $(K+1)^{st}$ retransmission is successful. The excess delay will be (K+1)(R + N + 1) - B and the relative excess delay will be $\frac{(K+1)(R+N+1) - B}{B}$. Thus, the relative excess delay can be bounded by:

$$0 \leq \frac{(K+1)(R + N + 1) - B}{B} < \frac{R + N + 1}{B} < 100\% \qquad (12)$$

Of course, this depends on the source having an accurate estimate of the RTD. For the poll-based scheme, the relative excess delay was given in (9) as $\frac{\delta + R + 2}{B}$, where δ lies between 0 and NI, assuming the first poll sent after the burst ends is successful. The drawback of the timer-based scheme during a long error burst is that the lost messages are unnecessarily retransmitted K times. This could be detrimental if the burst loss is due to congestion and window-based flow control is used (refer back to the discussion in Section 3.2).

5. POLL-BASED SCHEMES VS. TIMER-BASED SCHEMES

Let's review the major differences between poll-based and timer-based retransmission schemes. First, the performance of timer-based schemes is more heavily dependent upon the source's ability to estimate the RTD. Second, assuming messages travel in-sequence, a poll-based scheme can be implemented such that there are no unnecessary retransmissions. Third, the two schemes perform quite differently when bursts of messages are lost. If the burst loss is shorter than one round trip delay, then the excess delay in the poll-based scheme increases by about the duration of the burst; the excess delay in the timer-based scheme is relatively unaffected. If the burst loss is longer than one round trip delay, then the excess delay increases in both schemes, although the increase in poll-based schemes is likely to be greater; however, timer-based schemes trigger retransmissions too quickly in this scenario so that some retransmissions will be lost as part of the burst loss.

Using this analysis, we examine whether a timer-based scheme or a poll-based scheme is more appropriate for a given network. Throughout this section it is assumed that packets travel in order. In Section 5.3, we discuss schemes that include both a poll mechanism and a timer.

5.1 Low Speed Networks

First, let's consider a low speed system where the message size is small. Neither type of scheme performs very well in this environment. In poll-based schemes, the transmission time of the poll represents a significant portion of the total retransmission time. In timer-based schemes, the low speed of the network may result in highly variable queueing delays; the inability to properly estimate the RTD could potentially result in large excess delay or in many unnecessary retransmissions.

Since the major concern on low speed networks is likely to be bandwidth usage, poll-based schemes are preferred since they do not produce unnecessary retransmissions. Note that if a poll-based scheme is used, the polls should be piggybacked to reduce the polling overhead.

One could argue that if buffers at the intermediate nodes are kept very small, then the queueing delays would be less variable, enabling a timer-based scheme to perform better. However, small buffers would likely lead to bursts of messages being dropped due to congestion. Assuming window-based flow control is used, the extra retransmission delay in poll-based schemes during times of congestion would be desirable. Thus, even in this situation, we see that poll-based schemes are preferred.

For low speed networks where the message size is large, poll-based schemes provide small expected relative excess delay (for random losses), in addition to being able to prevent unnecessary retransmissions. Thus, their advantage over timer-based schemes is even more pronounced in this environment.

In general, we conclude that for low speed systems, poll-based schemes are preferable to timer-based schemes.

5.2 High Speed Networks

First, consider the case where all speeds in a network are scaled up by the same factor, while the packet size and buffer size remain fixed. As shown in Section 2 on Graph 2, poll-based schemes perform well at high data rates (assuming bursts of messages are not lost). Also, as the data rates increase, queueing delays decrease, assuming flow control remains equally effective. Thus, timer-based schemes also perform well. In general, both types of schemes should provide low expected relative excess delay when there are random losses if all connections are running at high speed.

High speed networks, however, are likely to carry a wide range of traffic types; the data rates and burstiness of the various connections may span a wide range. Thus, despite the fact

that the data links are running at high speed, some of the individual connections may be running at low speed. This mix of traffic favors the use of timer-based schemes. The high speed of the links will result in low queueing delay. Thus, the estimates of the RTD should be fairly accurate. For the low speed connections in the system, the delay in waiting for a poll to arrive may be relatively large (due to the transmission delay in sending a poll); thus, these connections may suffer a large relative excess delay if poll-based schemes are used.

Some type of rate-based flow control is likely to be used on these integrated service systems, which again favors the use of timer-based schemes. In rate-based flow control, a certain number of messages will be sent in a given time period, regardless of whether the messages are new transmissions or retransmissions. Thus, if the system is congested, delaying retransmissions does not help alleviate the congestion (unless an upper limit on the send window is reached). Thus, from the standpoint of delay, the retransmissions should be sent as quickly as possible, which favors the use of timer-based schemes.

One strategy that might be used on high speed networks is to make the buffers at the intermediate nodes very large. Flow control would still be implemented with the goal of keeping queueing delays small. However, if severe congestion develops, packets will be queued in buffers at the intermediate nodes rather than being dropped. The rationale for this is that queueing a message in a buffer will probably result in less overall delay than dropping the message and retransmitting it. In such a scheme, losses due to congestion would be less likely; also, the queueing delays may be more variable, especially if the flow-control mechanism is not effective. Thus, the performance difference between timer-based and poll-based schemes would be less significant in such an environment. However, if burst errors are expected to be common, then again the increase in excess delay which accompanies burst losses in poll-based schemes would be undesirable.

Overall, timer-based schemes are better able to provide low delay in a high speed, integrated service environment.

5.3 Combination of Polling and Timer Mechanism

It is possible to use both polls and a timer to generate retransmissions. The polling mechanism should be relied upon as the primary means of generating retransmissions since it does not produce spurious retransmissions. The timer mechanism should be used to provide an upper limit to the excess delay.

The polling portion of the scheme can be similar to the scheme described in Section 2, where the source sends polls and the destination responds with status messages that can trigger

retransmissions. In addition, whenever a message is received successfully at the destination, the destination sends an ACK of the message back to the source. The source maintains its retransmission timers based on the transmission times of polls rather than data. Assume poll P is transmitted at time $\mathcal{T}$, and assume the source's estimate of the RTD is R_E. The timer corresponding to poll P should be set to expire at time $\mathcal{T} + 2 + \beta R_E$, where $\beta > 1$ permits some error in estimating R_E and where 2 represents the transmission time of the poll and the corresponding status message. If the timer corresponding to poll P expires without the source receiving a status message corresponding to poll P or corresponding to a poll sent after poll P, the source retransmits all unacknowledged messages sent prior to poll P. (Another poll/timer scheme is described in [15].)

A combination poll/timer scheme is most appropriate for a network that has characteristics that are somewhere 'in between' low speed and high speed. The RTD should be somewhat varying such that relying on a timer scheme alone would produce too many retransmissions. However, the RTD should not be so difficult to estimate that β needs to be set very high in order to avoid many retransmissions (which essentially renders the timer mechanism almost useless). Low retransmission delay should be somewhat of a concern, to make it worthwhile to implement the timer scheme in order to upper bound the excess delay. Bandwidth efficiency should be somewhat of an important issue, otherwise there would be no need to implement polls.

The more closely the characteristics of the system resemble those of a high speed network, i.e., the better the source can estimate the RTD and the more important low delay is, the smaller β should be. The smaller β is, the larger role the timer mechanism plays. Conversely, the lower the speed of the system, the larger β should be; this increases the role of the polls and status messages. Implementing both polling and timer mechanisms allows the source a lot of flexibility in establishing the tradeoff between low delay and bandwidth efficiency. The drawback is the added complexity. The source has to deal with sending polls and setting timers, and the destination has to send ACKs as well as status messages.

6. CONCLUSION

Our analysis shows that the property of no unnecessary retransmissions makes poll-based retransmission schemes well suited to systems where bandwidth efficiency is important. The potential of large excess retransmission delay during burst losses is the chief drawback to using such schemes on high speed networks, where low latency is important. On high speed networks, the speed of the links should result in low round trip delay variability, which favors

the use of timer schemes. Another option is to combine the best features of the two schemes and use polls as the primary means of generating retransmissions and use a timer as a means of upper bounding the retransmission delay.

References

[1] Schwartz, M., *Telecommunication Networks*, Addison-Wesley Publishing Co., Reading, MA, 1987.

[2] Zhang, L., "Why TCP Timers Don't Work Well," *ACM SIGCOMM '86 Symposium*, Stowe, VT, Aug. 5-7, 1986, pp. 397-405.

[3] Comer, D., *Internetworking with TCP/IP Volume I*, Second Edition, Prentice- Hall, New Jersey, 1991.

[4] Doeringer, W., Dykeman, D., Kaiserwerth, M., Meister, B., Rudin, H., and Williamson, R., "A survey of light-weight transport protocols for high-speed networks," *IEEE Transactions on Communications*, 38:2025-2039, 1990.

[5] CCITT Study Group XI Temporary Document XI/2-24, "Results of September 1992 XI/2 discussions on Q.SAAL," Geneva, September, 1992.

[6] Protocol Engines, Inc., "XTP Protocol Definition," Revision 3.4, November, 1989.

[7] Sanders, R. and Weaver, A., "The Xpress Transfer Protocol (XTP) - A Tutorial," *Computer Communication Review*, vol. 20 no. 5, October 1990.

[8] Konheim, A., "A queueing analysis of two ARQ Protocols," *IEEE Transactions on Communications*, 28: 1004-1014, 1980.

[9] Rosberg, Z. and Shacham, N., "Resequencing delay and buffer occupancy under the selective-repeat ARQ," *IEEE Transactions on Information Theory*, 35:166-173, 1989.

[10] Anagnostou, M. and Protonotarios, E., "Performance Analysis of the Selective Repeat ARQ Protocol", *IEEE Transactions on Communications*, 34:127-135, 1986.

[11] Simmons, J., "Proof of correctness of proposed ATM retransmission scheme," submitted to *IEEE /ACM Transactions on Networking*, 1993.

[12] Bux, W., Kermani, P., and Kleinoeder, W., "Performance of an improved data link control protocol," *Proceedings ICCC'88*, Tel Aviv, Oct. 30-Nov. 3, 1988, pp. 251-258.

[13] Clark, D., Lambert, M., and Zhang, L., "NETBLT: A high throughput transport protocol," *ACM SIGCOMM '87 Symposium*, Stowe, VT, Aug., 1987, pp. 353-359.

[14] Feldmeier, D. and Biersack, E., "Comparison of error control protocols for high bandwidth-delay product networks," *Protocols for High Speed Networks II*, Marjory Johnson, Editor, Elsevier Science Publishers, 1991, pp. 271-295.

[15] Netravali, A., Roome, W., and Sabnani, K., "Design and implementation of a high-speed transport protocol," *IEEE Transactions on Communications*, 38:2010-2024, 1990.