LIDS-P-2191

PROOF OF CORRECTNESS OF PROPOSED ATM RETRANSMISSION SCHEME

August, 1993

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

Abstract: The CCITT has proposed a poll-based retransmission scheme as part of the Service Specific Connection Oriented Protocol for ATM systems. The basic scheme consists of the source periodically sending polls to the destination indicating which frames have been sent, and the destination responding with a status message indicating which of these frames have not been received. Many additional features have been included in the scheme in order to reduce the retransmission delay and prevent unnecessary retransmissions. With the added complexity of these features, it is not readily apparent whether the scheme generates the necessary retransmissions. We formally prove that the scheme does eventually generate a retransmission of any lost frame without producing any unnecessary retransmissions, assuming that certain reasonable conditions hold. In proving the correctness of the scheme, we also further define the protocol. We also examine how the protocol can fail if the conditions for proper operation are not met.

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

1. INTRODUCTION

The ATM Adaptation Layer is comprised of two sublayers: the lower portion is referred to as the 'common part' and the higher portion is referred to as the 'service specific part'[1]. There are several protocols that have been proposed for the common part, e.g., AAL Type 5 is proposed as a common part protocol for connection oriented traffic.[2] One function of the common part is to detect corrupt frames. If assured data transfer is desired, then it is the responsibility of the service specific protocol to ensure that any lost or corrupt frames are retransmitted. The CCITT has proposed a poll-based retransmission scheme as part of the Service Specific Connection Oriented Protocol (SSCOP) for ATM.[1] Retransmissions are performed end-to-end. The unit of retransmission is properly referred to as a Sequenced Data Protocol Data Unit; for simplicity we will refer to the unit of retransmission as a frame.

The basic scheme consists of the source periodically sending polls to the destination indicating which frames have been sent, and the destination responding with a status message indicating which of these frames have not been received. However, many additional features have been included in the proposed scheme. For example, in order to reduce retransmission delay, the scheme takes advantage of the fact that frames are expected to travel in sequence in ATM systems. If the destination receives frame X without having received frame X-1, it immediately sends a NACK of frame X, rather than waiting for a poll. Also, a major design goal was to ensure that the scheme does not produce any unnecessary retransmissions. In order to accomplish this, the source and destination must maintain several variables. With the added complexity of these features, it is not readily apparent whether the scheme generates the necessary retransmissions.

Further details of the scheme are provided in Section 2. In Section 3, we formally prove that the scheme does eventually generate a retransmission of any lost frame without producing any unnecessary retransmissions, assuming that certain reasonable conditions hold. In proving the correctness of the scheme, we also further define the protocol. In Section 4, we examine how the protocol can fail if the conditions for proper operation are not met.

In this paper, we only address the issue of whether the scheme works, not whether the scheme is appropriate for the ATM environment. For an analysis of poll-based retransmission schemes in terms of delay and overhead, refer to [3].

2. DESCRIPTION OF PROPOSED RETRANSMISSION SCHEME

First, we provide an overview of the proposed retransmission scheme. For complete details of the proposed scheme, see [1,4]. In section 2.2, we transform the protocol definition into algorithmic form. An example is provided in section 2.3 that illustrates many of the details of the protocol.

2.1 Overview of Proposed Scheme

Each data frame in a connection is numbered sequentially modulo 2^{24}. Also, each poll message is numbered sequentially modulo 2^{24}, independently of the data frames. The source maintains the variable SEQ to indicate that all data frames up to but not including SEQ have been transmitted at least once, and the variable PSEQ to indicate the sequence number of the last poll message transmitted. After a data frame is transmitted, the frame is stored in a retransmission buffer, along with the current value of PSEQ. Polls are sent periodically by the source.

There are two types of status messages sent by the receiver. A solicited STAT is sent in response to a poll message and provides the status of all frames through SEQ-1, where SEQ is indicated in the poll message. It also includes the value of PSEQ contained in the incoming poll message.

The receiver sends an unsolicited STAT whenever it determines that there are frames missing. An unsolicited STAT only NACKs frames that have not previously been NACKed in another status message. The one exception to this is that the protocol provides the option for the receiver to send two identical unsolicited STATs rather than just one.

When the source receives a solicited status message, it retransmits all NACKed frames as long as the value of PSEQ stored along with the frame in the retransmission buffer is less than the value of PSEQ indicated in the status message. Essentially the retransmission criterion is: retransmit any NACKed frames that were sent before the poll that triggered the NACK.

There is a logical variable stored with each transmitted data frame, called RTS, that indicates whether the frame has ever been retransmitted due to an unsolicited STAT. When the source receives an unsolicited status message, it retransmits all NACKed frames as long as the corresponding RTS value is FALSE. No comparisons of PSEQ are done.

2.2 Source and Destination Algorithms

There are several variables that must be maintained by the source and destination, as listed below.[1] In this section, we assume all variables and sequence numbers are ordinary integers, rather than integers modulo 2^{24}.

Source Variables

SEQA = oldest transmitted but unacknowledged frame at source

SEQ = one greater than highest numbered frame transmitted by source

---

[1] The most recent proposal uses different variable names than those used below. The mapping between the variable names is:
SEQ = VT(S) PSEQ = VT(PS) SEQA = VT(A) PSEQL = VT(PA)-1 SEQR = VR(R) SEQH = VR(H)

PSEQ = sequence number of poll most recently sent by source

PSEQL = poll sequence number contained in solicited STAT most recently received by source

RTS = logical variable associated with each frame transmitted by source. It is initialized to FALSE and set to TRUE if the frame is retransmitted due to an unsolicited STAT

Destination Variables

SEQR = lowest consecutive frame that the destination hasn't correctly received

SEQH = one greater than the highest numbered frame that the destination knows the source has sent

We also use the following conventions:

$PSEQ_P$ = PSEQ number sent in a poll

SEQ_P = SEQ number sent in a poll

$PSEQ_S$ = PSEQ number contained in a solicited STAT

$SEQR_S$ = SEQR number contained in a STAT (either solicited or unsolicited STAT)

P_X = PSEQ stamp of frame number X (stored in retransmission buffer)

Algorithm at Source

1. Initialize SEQA, SEQ, PSEQ, and PSEQL to 0. The first transmitted data frame contains sequence number 0. The first transmitted poll contains poll sequence number 1.
2. Do steps 3 through 8 repeatedly. Steps 5 through 8 are performed whenever the conditions are met. Steps 3 and 4 are done repeatedly within finite intervals chosen by the source.
3. If SEQ < SEQA + 2^{24} - 1, and a frame is available from the higher layer, assign frame number SEQ to the frame, and increment SEQ by one (this may not be possible due to the flow control window). Transmit frame, and store frame in retransmission buffer with current value of PSEQ, and set the corresponding RTS variable to FALSE.
4. If PSEQ < PSEQL + 2^{24} -1, increment PSEQ by one. Send a poll containing poll number PSEQ and the current value of SEQ.
5. If a STAT is received with $SEQR_S$ > SEQA, increase SEQA to $SEQR_S$.
6. When a solicited STAT is received, set PSEQL to $PSEQ_S$.
7. If a NACK of frame number X is received in a solicited STAT containing $PSEQ_S$, then retransmit frame X if P_X < $PSEQ_S$. If frame X is retransmitted, update P_X to current value of PSEQ.
8. If a NACK of frame number X is received in an unsolicited STAT, retransmit X if the corresponding RTS variable equals FALSE. If frame X is retransmitted, set P_X to current value of PSEQ, and set RTS to TRUE.

Algorithm at Destination

1. Initialize SEQR and SEQH to 0. Do steps 2, 3, and 4 repeatedly.

2. If receive a frame with sequence number X equal to SEQR, accept the frame, and increment SEQR. Perform the following loop:

    While (SEQR equals the sequence number of a frame already in resequencing buffer) {
        increment SEQR }

Pass up to the higher layer all frames from X through the new value of SEQR-1 (in proper sequence). If $X \geq SEQH$, update SEQH to X + 1.

3. If receive a frame with sequence number X such that $SEQR < X \leq SEQR + 2^{24} - 2$, store the frame in the resequencing buffer. (The receive window may actually be smaller due to flow control restrictions.) If X > SEQH, NACK all missing frames between SEQH and X-1, inclusive (if any), in an unsolicited STAT. Set $SEQR_S$ in the STAT to the current value of SEQR. As an option, two identical unsolicited STATs can be sent. If $X \geq SEQH$, update SEQH to X + 1.

4. If receive a poll with $SEQ_P > SEQH$, optionally NACK all missing frames between SEQH and $SEQ_P - 1$, inclusive (if any), in an unsolicited STAT. Set $SEQR_S$ in the STAT to the current value of SEQR. Also, if $SEQ_P > SEQH$, update SEQH to SEQ_P. Regardless of whether the unsolicited STAT is sent, send a solicited STAT containing $PSEQ_S$ equal to $PSEQ_P$, containing $SEQR_S$ equal to the current value of SEQR, and NACKing all missing frames with sequence number less than SEQ_P.

Throughout our discussion, it is assumed that each of the steps in the above algorithms at the source and destination can be viewed as an indivisible operation i.e., once a given step is started, no other operations are performed until the entire step is finished. When we refer to transmissions at the source and destination, we assume the frame or control message is passed down to a transmit queue at a lower layer, which is served in First In First Out order.

2.3 Example of Operation

An example of the polling scheme operation is shown in Figure 1. The important points are described below. In the early versions of the proposal, both selective repeat and Go Back N operation were included, but in the most recent proposal, it appears just selective repeat is supported. (For a discussion of Go Back N and selective repeat, see [5,6].)

In the example, frames 1 and 2 are delivered successfully. Frame 3 is lost, and Poll 1 generates a solicited STAT that NACKs frame 3. (We assume the receiver chooses not to also send an unsolicited STAT in response to the poll.) Note that the arrival of the poll also causes SEQH to be updated to 4 (i.e., due to the poll, the destination knows that all frames through 3

have been sent at least once). Frame 4 is lost; frame 5 is received successfully and triggers an unsolicited STAT that NACKs frame 4.

The PSEQ value stored with frame 3 is 0; thus, solicited STAT 1 triggers the retransmission of frame 3 (since $0 < 1$). The unsolicited STAT of frame 4 automatically triggers the retransmission of frame 4 without any PSEQ comparisons being necessary.

Before the retransmissions of frames 3 and 4 are sent, Poll 2 is sent, which generates a solicited STAT that NACKs both frames 3 and 4. Solicited STAT 2 triggers no retransmissions since the PSEQ value stored with frames 3 and 4 is now 2. The retransmission of frame 3 is lost, but the retransmission of frame 4 is successful. Even though the destination receives frame 4 and not frame 3, it does not send an unsolicited STAT, since SEQH is 6. Finally, Poll 3 generates solicited STAT 3 that NACKs frame 3. This triggers the successful retransmission of frame 3.

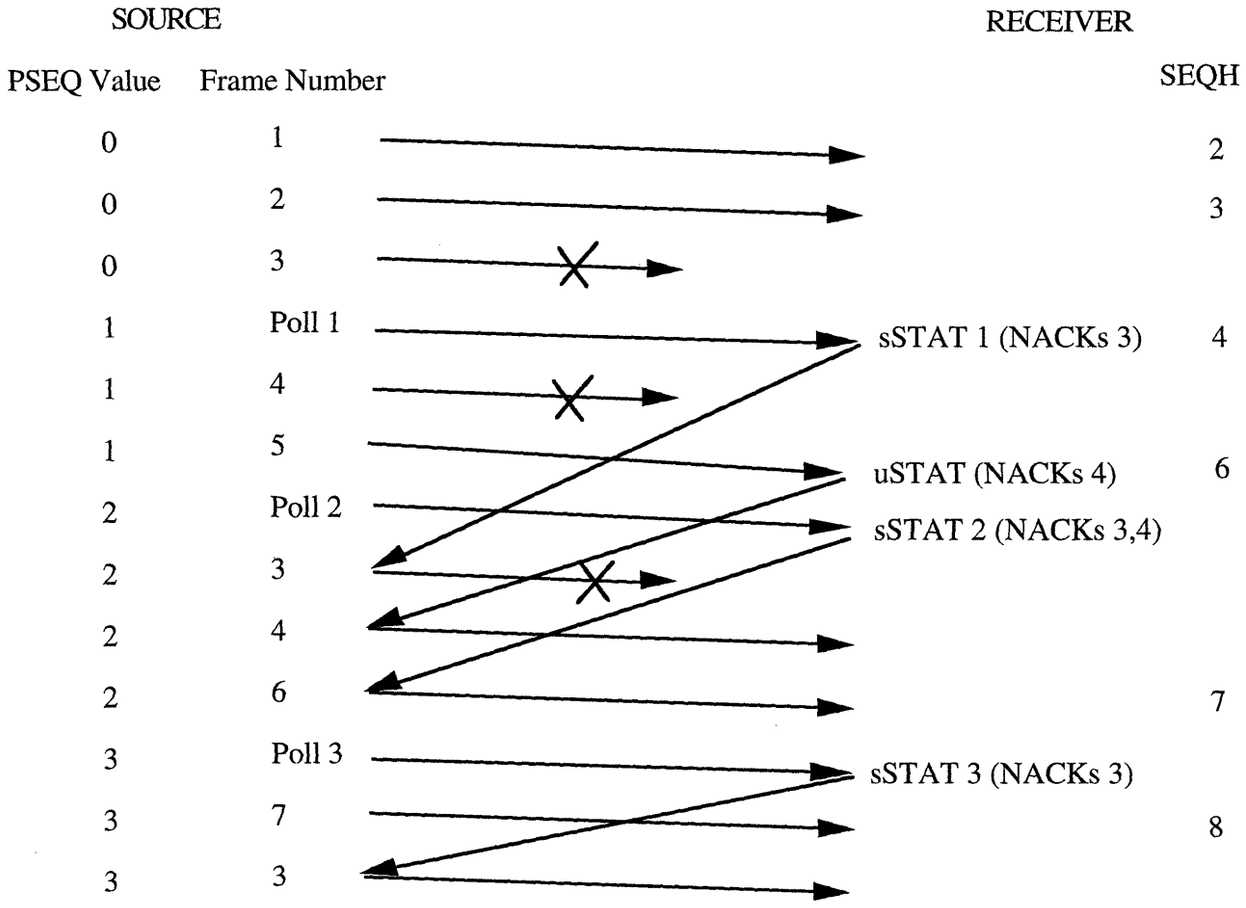

Figure 1 Operation of ATM retransmission scheme.

3. PROOF OF PROPER OPERATION

As can be seen from the previous section, the proposed ATM retransmission algorithm requires the maintenance of several variables, and involves many special cases. The goal of this section is to prove the correctness of the protocol. We must show that the source can continue forever to accept frames for transmission from the higher layer, and that all frames are eventually delivered in proper sequence at the destination. We also need to show that the claim of no unnecessary retransmissions is true.

3.1 Conditions of Normal Operation

It is easy to come up with special circumstances where the proposed ATM retransmission scheme does not work correctly. Thus, we first must define the conditions of "normal operation". Under these conditions, we can prove the correctness of the protocol. The conditions of normal operation are:

1) Frames (data or control) travel in order on the links.
2) Undetected errors do not occur.
3) State information at the source and receiver is not lost.
4) The connection does not go down.
5) There is some $q > 0$ such that each frame (data or control) is received error-free with probability at least q.
6) All transmission and propagation delays are finite.
7) If polls are numbered modulo 2^{24}, then no more than $2^{24}-2$ consecutive poll/status message combinations are lost (i.e., one out of every $2^{24}-1$ consecutive polls arrives successfully at the destination, and the corresponding status message arrives successfully at the source).
8) If frames are numbered modulo 2^{24}, and the oldest unacknowledged frame at the source is SEQA, then the source does not transmit past frame SEQA + 2^{24} - 2. (Due to the flow control window, the upper limit of the send window may be much smaller than this.)
9) If polls are numbered modulo 2^{24}, and the solicited STAT most recently received by the source contained a poll sequence number of PSEQL, then the source does not transmit past poll number PSEQL + 2^{24} - 1.

The first seven conditions deal with properties of the network that must hold to guarantee proper operation. The last two conditions should really be specified as part of the protocol. Also, we make the obvious assumption that a frame cannot be received before it is sent. We also assume that in the initial state of the system, there are no frames on any of the links.

Below, we prove the proper operation of the protocol given the above conditions. We first prove the protocol works if all sequence numbers are integers that can increase without bound. In Section 3.5, we consider the case where frame sequence numbers and poll sequence numbers are integers modulo 2^{24}. The methodology of the proof closely follows that used in [6] to prove the correctness of general Go Back N schemes. To prove the correctness of the protocol, we must show *safety* and *liveness*.

3.2 Safety Condition

The algorithm is safe if it never delivers an out-of-sequence frame to the higher layer at the destination. From step 2 of the algorithm at the destination, we see that frames must be passed up in sequence, so that the algorithm is safe.

3.3 Liveness Condition

The algorithm is live if the source can continue forever to accept packets from the higher layer, and the destination continues to deliver them to the higher layer.

Let SEQA(t), SEQ(t), PSEQ(t), SEQR(t), and SEQH(t) represent the value of these five variables at time t. From the algorithm statement, it can be seen that all five must be non-decreasing in t. Define a **successful poll** as a poll that gets to the destination error-free, and whose corresponding solicited STAT gets to the source error-free.

Consider any transmitted frame with sequence number X. Refer to Figure 2. Let t_{Ti} equal the transmission time of the i^{th} copy of frame X. Polls are sent periodically within finite intervals; due to condition 5 above, some poll sent after time t_{Ti} must be successful. Let t_{Pi} be the transmission time of the first successful poll sent after time t_{Ti}. Let t_{Ri} be the time this poll arrives at the destination. Let t_{Si} equal the time the STAT corresponding to this poll arrives at the source. Thus, $t_{Ti} < t_{Pi} \leq t_{Ri} \leq t_{Si}$. Since we assume finite delays, and because of condition 5, t_{Ri}, and t_{Si} are finite.

Let $P_X(t)$ represent the PSEQ stamp of frame X at time t. Then

$$P_X(t) = PSEQ(t_{Ti}) \quad \text{for } t_{Ti} \leq t < t_{T(i+1)} \qquad (1)$$

where we take $t_{T(i+1)}$ to be ∞ if X is not transmitted for the $(i+1)^{th}$ time.

Also, since SEQ is one greater than any transmitted frame:

$$SEQ(t) > X \quad \text{for } t > t_{Ti} \qquad (2)$$

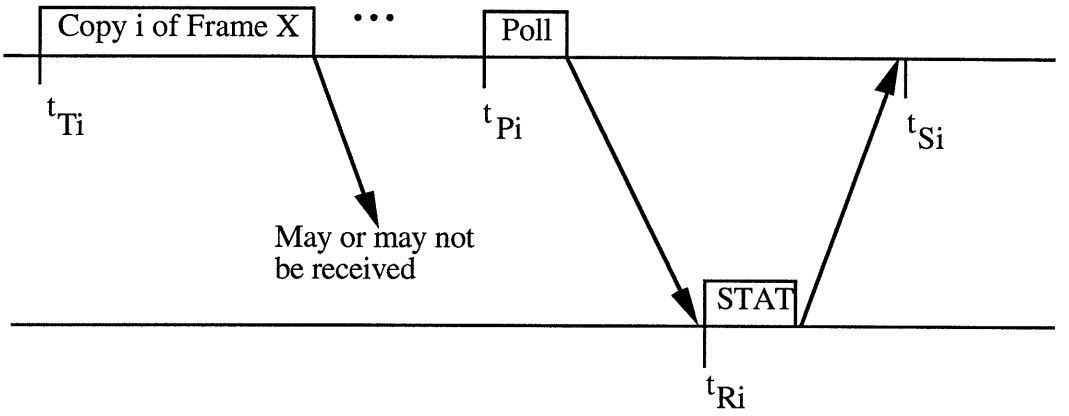

Figure 2 Copy i of frame X is transmitted at time t_{Ti}. Due to condition 5, some poll that it is transmitted after time t_{Ti} will be successful. We assume such a poll is sent at time t_{Pi} and received at the destination at time t_{Ri}. The corresponding solicited status message is received at the source at time t_{Si}.

We want to show that if the i^{th} copy of frame X has been transmitted, either this copy will be received successfully at the destination, or copy i+1 of frame X will be transmitted. (In the next section, we show that both events do not occur.)

The successful poll sent at time t_{Pi} contains $PSEQ_P = PSEQ(t_{Pi}^-) + 1$ and $SEQ_P = SEQ(t_{Pi}^-)$, where t_{Pi}^- represents the start of the poll transmission operation, before PSEQ is updated. Since $t_{Pi}^- > t_{Ti}$, from (2), we have:

$$SEQ_P > X \qquad (3)$$

Also, the nondecreasing property of PSEQ(t) implies that:

$$PSEQ_P > PSEQ(t_{Ti}) \qquad (4)$$

At time t_{Ri}, one of the following 2 conditions must hold:

a) Frame X has been received by the destination. Since t_{Ri} is finite, this implies frame X has been received in finite time.

b) Frame X has not been received by the destination. Since the poll was sent after frame X, and it arrives before frame X, it means frame X is lost (since frames travel in order). The poll triggers a solicited STAT NACKing frame X since, as shown above, $SEQ_P > X$. The solicited STAT, with $PSEQ_S$ equal to $PSEQ_P$, arrives at the source at time t_{Si}. It is possible an unsolicited STAT NACKing copy i of frame X has already been received by the source prior to t_{Si}, in which case copy i+1 may already have been sent. If not, then, from equation (1), $P_X(t_{Si}) = PSEQ(t_{Ti})$. Combining this with equation (4) yields:

$$P_X(t_{Si}) = PSEQ(t_{Ti}) < PSEQ_P = PSEQ_S$$

Thus, since $P_X(t_{Si}) < PSEQ_S$, the condition of step 7 in the algorithm at the source is satisfied, and frame X is retransmitted.

We conclude that, given copy i of frame X has been sent, either copy i will be successfully received, or copy i+1 will be transmitted. From condition 5, we assume that frames are successfully received with probability greater than some non-zero q. Thus, eventually frame X will be received successfully. (Note that even without unsolicited STATs, frame X is eventually received successfully; only solicited STATs are needed to satisfy liveness.)

Now, let Y be the value of SEQA at any time t. We know that all frames before Y have been received at the destination and have been ACKed; thus, frame Y must fall within the receive window. From what was shown above, frame Y will eventually be received at the destination in finite time. At the time Y is accepted at the destination, SEQR will be incremented beyond Y. The destination will be able to deliver to the higher layer all frames through at least frame Y. The next successful poll sent after the successful transmission of Y will generate a solicited STAT with $SEQR_S > Y$. Thus, at the time this STAT arrives at the source, SEQA will be incremented beyond Y.

We conclude that the value of SEQA is always incremented after some finite time (assuming there is data to send). This allows the higher layer at the source to continue to submit frames.

We have shown that the algorithm is live: the higher layer at the source can continue to submit frames, and the destination will continue to deliver them. Next, we need to show the algorithm does not produce any unnecessary retransmissions.

3.4 No Unnecessary Retransmissions

We define an unnecessary retransmission as occurring when copy j of a frame is sent even though copy i, for some $i < j$, is not lost. If copy i+1 is not sent unless copy i is lost, then we know that copy j, for all $j > i$, will not be sent unless copy i is lost. Thus, we just need to consider copies i and i+1.

In the discussion below, we examine whether copy i+1 of an arbitrary frame, say frame X, could ever be unnecessarily sent. In order for copy i+1 to be unnecessary it must be true that copy i of frame X does arrive (and is accepted) at the destination.

There are 3 possible ways a NACK of frame X can be generated (for simplicity, SEQH is used rather than SEQH(t)):

1) Frame Y arrives at the destination, and at the time of its arrival X has not arrived and been accepted, and the following holds: $Y > X \geq SEQH$. An unsolicited NACK of frame X is sent and SEQH is updated to Y+1.

2) A poll arrives at the destination, and at the time of its arrival X has not arrived and been accepted, and the following holds: $SEQ_P > X \geq SEQH$. An unsolicited NACK of frame X is

optionally sent. A solicited NACK of frame X must be sent. After the NACKs are sent, SEQH is updated to SEQ_P.

3) A poll arrives at the destination, and at the time of its arrival X has not arrived and been accepted, and the following holds: $SEQ_P > X$ and $SEQH > X$. A solicited NACK of frame X is sent. If $SEQ_P > SEQH$, then after the NACK is sent, SEQH is updated to SEQ_P.

First, consider the case where $i = 1$. Assume copy 1 of frame X arrives (and is accepted) at the destination. Copy 1 of frame X must be sent before any copy of frame Y, where $Y > X$, and must be sent before a poll with SEQ_P, where $SEQ_P > X$. Given that frames travel in order, such a frame Y or such a poll cannot arrive before frame X. Thus, the conditions in the three procedures above are not satisfied; copy 1 of frame X will not be NACKed, so no further copies of frame X will be sent.

Next, consider the case where $i > 1$. Since frame X has been sent more than once, and since frames are only retransmitted in response to NACKs, it must be true that frame X was NACKed at least once. From statements 1, 2, and 3 above, it must be true that after frame X is NACKed, SEQH is updated to a value greater than X. (In statement 3, SEQH is already greater than X.) SEQH is nondecreasing. Thus, after frame X has been NACKed once, statements 1 and 2 above can never be satisfied, since they require $X \geq SEQH$ (i.e., an unsolicited STAT can only NACK copy 1 of a frame.) (For now, ignore the case where two identical unsolicited STATs are sent.)

Thus, we only need to consider statement 3 above. Let t_i be the time that copy i of frame X is transmitted. Let P_{Xi} be the PSEQ stamp associated with copy i of frame X. Then $P_{Xi} = PSEQ(t_i)$. We assume that copy i is received and accepted by the destination. We consider whether any poll can trigger an unnecessary transmission of copy i+1 of frame X.

First, consider a poll sent before t_i. Its poll sequence number, $PSEQ_P$, satisfies $PSEQ_P \leq PSEQ(t_i)$. Thus, the corresponding solicited STAT would contain $PSEQ_S \leq PSEQ(t_i) = P_{Xi}$. Thus, copy i+1 of frame X would not be transmitted since P_{Xi} is not less than $PSEQ_S$.

Next, consider a poll sent after t_i. If copy i of frame X gets to the destination, then it must arrive before such a poll (since frames travel in sequence). Thus, this poll will not NACK frame X.

Thus, a poll will not trigger an unnecessary retransmission of frame X.

The last case we need to consider is where copy 1 of frame X is lost, and frame X is retransmitted due to an unsolicited STAT. The destination has the option of sending two identical unsolicited STATs. However, if frame X is retransmitted due to an unsolicited STAT,

the variable RTS_X is set to TRUE, so that all future NACKs of frame X contained in unsolicited STATs are ignored. Thus, if both unsolicited STATs arrive at the source, the second one will be ignored.

Overall, we see that given copy i of frame X is received, copy i+1 will not be sent. Thus, no unnecessary retransmissions are produced.

3.5 Sequence Numbers with Modulus

In the sections above, we showed that the protocol works correctly if sequence numbers can increase without bound. In this section, we show that the protocol continues to work if frame sequence numbers and poll sequence numbers are treated modulo 2^{24}. We must check that the acceptance policy, the NACK procedure, and retransmission procedure are unaffected if the modulus is used.

3.5.1 Received Frame Sequence Numbers

First, continue to assume that sequence numbers are integers increasing without bound. Consider the successful transmission of an arbitrary frame sent by the source. Assume it is transmitted at time t_1 and received at the destination at time t_2. Obviously, $t_2 \geq t_1$. The sequence number of the frame, say X, must lie in the source's send window at time t_1. Thus:

$$SEQA(t_1) \leq X \leq SEQA(t_1) + 2^{24} - 2 \qquad (5)$$

From step 5 of the algorithm at the source, it must be true that $SEQA(t) \leq SEQR(t)$ for all t. (If $SEQA(t) > SEQR(t)$, it would mean the source received an ACK for a frame that was not received by the destination.) Thus, for any $t_1 \leq t_2$,

$$SEQA(t_1) \leq SEQR(t_1) \leq SEQR(t_2) \qquad (6)$$

We showed above that the protocol does not produce unnecessary retransmissions. Thus, the transmission of frame X must be necessary. Thus, $SEQR(t_2) \leq X$. (If $SEQR(t_2)$ were greater than X, it would mean frame X has already been received by time t_2.) Combining this with equations (5) and (6), yields:

$$SEQA(t_1) \leq SEQR(t_2) \leq X \leq SEQA(t_1) + 2^{24} - 2 \qquad (7)$$

The destination accepts frames in the range from $SEQR(t_2)$ to $SEQR(t_2) + 2^{24} - 2$. (The window of acceptance may actually be smaller for flow control purposes.) From (7) we have: $0 \leq (X - SEQR(t_2)) \leq 2^{24} - 2$. Thus, X falling in the range $SEQR(t_2)$ to $SEQR(t_2) + 2^{24} - 2$ is equivalent to X mod 2^{24} falling in the range $SEQR(t_2)$ mod 2^{24} to $(SEQR(t_2) + 2^{24} - 2)$ mod 2^{24}. Thus, treating the sequence numbers as integers modulo 2^{24} does not affect the acceptance policy at the destination.

Note that we need to define what it means to "fall in the range" of A and B, where A and B are numbers modulo 2^{24}. One can envision the numbers from 0 to $2^{24}-1$ on a circle, increasing

clockwise. Then for any two numbers A and B, we define the region that extends clockwise from A to B as representing the numbers that fall between A and B.

The property of no unnecessary retransmissions is important in the above argument. In general selective repeat systems, where unnecessary retransmissions can occur, if the modulus is M, then ambiguity can occur if the source transmits past SEQA+M/2 - 1 (as opposed to SEQA + M - 2 for the ATM scheme).[6]

3.5.2 Generating Solicited Status Messages

Again, assume increasing integers are used for sequence numbers. Assume a poll is transmitted at time t_1, and arrives at the destination at time t_2. Obviously, $t_1 \leq t_2$. Assume the poll contains SEQ_P, where $SEQ_P = SEQ(t_1)$. Thus, at time t_1, all frames through $SEQ_P - 1$ have been transmitted. Due to the restriction on the send window, we know that:

$$SEQ_P - 1 \leq SEQA(t_1) + 2^{24} - 2 \qquad (8)$$

Since $SEQ(t) \geq SEQA(t)$ for all t, we have:

$$SEQA(t_1) \leq SEQ(t_1) = SEQ_P \leq SEQA(t_1) + 2^{24} - 1 \qquad (9)$$

In equation (6), we showed $SEQA(t_1) \leq SEQR(t_2)$ for any $t_1 \leq t_2$. The value of $SEQR(t_2)$ indicates that the source must have sent the frame with sequence number $SEQR(t_2) - 1$ at some time prior to t_1, say at time t_0. (If the frame were sent after t_1, it could not have arrived prior to the poll that was sent at t_1.) Using the restriction on the send window at the source, $SEQR(t_2) - 1 \leq SEQA(t_0) + 2^{24} - 2$. Since SEQA(t) is non-decreasing in t, we arrive at:

$$SEQA(t_1) \leq SEQR(t_2) \leq SEQA(t_1) + 2^{24} - 1 \qquad (10)$$

We have shown that both SEQ_P and $SEQR(t_2)$ lie between $SEQA(t_1)$ and $SEQA(t_1) + 2^{24} - 1$. A poll arriving at time t_2 generates NACKs of frames between $SEQR(t_2)$ and $SEQ_P - 1$, inclusive. Thus, sequence numbers can be treated modulo 2^{24} without causing ambiguity with NACKing frames in solicited STATs.

3.5.3 Generating Unsolicited Status Messages

Again, assume increasing integers are used for sequence numbers. Unsolicited STATs can be generated by the arrival of a frame or a poll. An unsolicited STAT is always sent if a frame with sequence number X arrives at time t before a frame with sequence number between SEQH(t) and X-1, inclusive (where SEQH(t) is the value of SEQH before it is updated due to the arrival at time t). An unsolicited STAT is optionally sent if a poll containing SEQ_P arrives at time t before a frame with sequence number between SEQH(t) and $SEQ_P - 1$, inclusive. Let t_1 be the transmission time at the source of the frame or poll that generates the unsolicited STAT, and let t_2 be the arrival time of the frame or poll at the destination.

From the definition of the protocol, it must be true that SEQR(t) ≤ SEQH(t) for all t. Thus, using equation (6), we know $SEQA(t_1) \leq SEQR(t_2) \leq SEQH(t_2)$. The value of $SEQH(t_2)$ indicates that the source must have sent the frame with sequence number $SEQH(t_2)$ -1 at some time prior to t_1, say at time t_0. Using the restriction on the send window at the source, $SEQH(t_2) - 1 \leq SEQA(t_0) + 2^{24} - 2$. Since SEQA(t) is non-decreasing in t, we arrive at:

$$SEQA(t_1) \leq SEQH(t_2) \leq SEQA(t_1) + 2^{24} - 1 \qquad (11)$$

From the restriction on the send window, we know that if frame X is sent at time t_1, then $SEQA(t_1) \leq X \leq SEQA(t_1) + 2^{24} - 2$. Or, if a poll with SEQ_P is sent at time t_1, then $SEQA(t_1) \leq SEQ_P - 1 \leq SEQA(t_1) + 2^{24} - 2$. Thus, using the rule for generating NACKs, any frame Y NACKed in an unsolicited STAT satisfies:

$$SEQA(t_1) \leq SEQH(t_2) \leq Y \leq SEQA(t_1) + 2^{24} - 2 \qquad (12)$$

Therefore, sequence numbers can be treated modulo 2^{24} without causing ambiguity with frames NACKed by unsolicited STATs.

3.5.4 Receiving Status Messages at Source

Again, assume increasing integers are used for sequence numbers. Let t_1 be the time a STAT (either solicited or unsolicited) is sent by the destination, and let t_2 be the time the STAT arrives at the source. Let $SEQR_S$ be the value of SEQR contained in the STAT. Let $SEQA(t_2)$ be the value of SEQA at the time the STAT arrives (i.e., before SEQA is updated to $SEQR_S$). Thus, all frames through $SEQR_S - 1$ are being ACKed by this STAT.

A STAT ACKing frame $SEQR_S$ cannot be received before a STAT that ACKs only through $SEQR_S - 1$ (since SEQR(t) is non-decreasing in t and frames travel in sequence). Thus, $SEQA(t_2) \leq SEQR_S$. (If $SEQA(t_2)$ were greater than $SEQR_S$, it would mean that $SEQR_S$ had been ACKed by time t_2.) Also, a STAT cannot ACK a frame that has not been sent. Due to the restriction on the send window, $SEQR_S - 1 \leq SEQA(t_2) + 2^{24} - 2$. Thus:

$$SEQA(t_2) \leq SEQR_S \leq SEQA(t_2) + 2^{24} - 1 \quad (13)$$

Thus, SEQA can be updated to $SEQR_S$ without ambiguity.

Now consider any NACKs contained in the STATs, and consider the sequence numbers as ordinary integers again. Let Y equal the sequence number of a NACKed frame in the STAT. By definition of the protocol, we know $Y \geq SEQR_S$.

First, consider the case where the STAT is an unsolicited STAT triggered by the arrival at the destination of a frame with sequence number X. By definition of the protocol, Y < X. We assumed the STAT is generated at time t_1; thus, we know that frame X has been sent by the source prior to t_1, say at time t_0. Due to the restriction on the send window, we know that: $X \leq SEQA(t_0) + 2^{24} - 2$. Thus, using the fact that SEQA(t) is non-decreasing in t:

$$Y < X \leq SEQA(t_0) + 2^{24} - 2 \leq SEQA(t_2) + 2^{24} - 2 \qquad (14)$$

Combining this with equation (13) and the fact that $Y \geq SEQR_S$ yields:

$$SEQA(t_2) \leq Y \leq SEQA(t_2) + 2^{24} - 2 \qquad (15)$$

Next, consider the case where the STAT is triggered by a poll containing SEQ_P (the STAT can be solicited or unsolicited). By definition of the protocol, $Y < SEQ_P$. The STAT is generated at time t_1; thus, we know that a poll containing SEQ_P has been sent by the source prior to t_1. Thus, we also know that frame $SEQ_P - 1$ must have been sent by the source before t_1, say at time t_0. Using the same argument as above again yields:

$$SEQA(t_2) \leq Y \leq SEQA(t_2) + 2^{24} - 2$$

Thus, at the arrival time t_2 of any STAT, $SEQR_S$ and any NACK contained in the STAT fall between $SEQA(t_2)$ and $SEQA(t_2) + 2^{24} - 2$.

Combining the last four sections, we see that all frame sequence numbers and SEQA, SEQ, SEQR, and SEQH can be kept modulo 2^{24} without affecting the operation of the protocol.

3.5.5 Poll Sequence Numbers

First consider poll sequence numbers as ordinary increasing integers. Let t_1 be the time a poll is transmitted by the source. Assume the poll contains $PSEQ_P$. Let t_2 be the time the corresponding solicited STAT gets back to the source (if it does). The STAT carries $PSEQ_S$ equal to $PSEQ_P$. PSEQL(t) represents the poll sequence number contained in the last received solicited STAT as a function of time. PSEQL(t) is non-decreasing in t. Since the STAT carrying sequence number $PSEQ_S$ does not arrive until time t_2, and since frames travel in sequence, $PSEQL(t_2) < PSEQ_S$ (we assume $PSEQL(t_2)$ represents the value of PSEQL at the arrival time of the STAT, before it is updated to $PSEQ_S$). At time t_1, $PSEQ_P$ must fall within the send window for polls. Thus, $PSEQ_P \leq PSEQL(t_1) + 2^{24} - 1$. Since PSEQL(t) is non-decreasing in t, $PSEQ_P \leq PSEQL(t_2) + 2^{24} - 1$. Thus, overall, we have:

$$PSEQL(t_2) < PSEQ_S = PSEQ_P \leq PSEQL(t_2) + 2^{24} - 1 \qquad (16)$$

Thus, any solicited STAT that arrives at the source at time t contains a PSEQ value that is within the range from PSEQL(t)+1 to $PSEQL(t) + 2^{24} - 1$, inclusive.

Now, let's consider the PSEQ stamps in the retransmission buffer. Consider any time t_2 when a solicited STAT arrives at the source. Obviously, the poll with $PSEQ_P$ equal to $PSEQ_S$ was sent before time t_2. Thus, $PSEQ(t_2) \geq PSEQ_P = PSEQ_S$.

At time t_2, the source examines each frame in the retransmission buffer:

- if a frame is ACKed by the STAT, the source removes it from the buffer
- if a frame is NACKed by the STAT and it is retransmitted then the PSEQ stamp of the frame is updated to $PSEQ(t_2)$.
- if a frame is NACKed by the STAT but it is not retransmitted, then the PSEQ stamp of the frame must have been greater than or equal to $PSEQ_S$.

• if a frame is in the buffer but not ACKed or NACKed by the STAT, then it must have been sent after the poll carrying $PSEQ_P$. Thus, the PSEQ stamp must be greater than or equal to $PSEQ_S$.

After the source finishes servicing the STAT, PSEQL is updated to $PSEQ_S$. From the above discussion we see that all frames in the retransmission buffer at this time will have a stamp greater than or equal to $PSEQ_S$. Thus, due to the restriction on the poll send window, after the STAT is serviced, all PSEQ stamps in the retransmission buffer lie between PSEQL and PSEQL+ 2^{24}-1, inclusive.

Thus, the poll sequence numbers can be treated modulo 2^{24} without ambiguity problems.

3.6 Out-of-Sequence Retransmissions

From the discussion above, it would seem that for any two frames, X and X+1, copy i of frame X+1 is never sent before copy i of frame X. However, this is not true even if the conditions stated in Section 2 hold. Consider the following example, which is depicted in Figure 3. Assume the destination sends a solicited STAT NACKing frame X, and later sends an unsolicited STAT NACKing frame X+1. The unsolicited STAT will not NACK frame X since unsolicited STATs can only NACK frames that have not been NACKed previously. Assume the solicited STAT is lost. When the unsolicited STAT arrives, frames X+1 will be retransmitted before frame X is retransmitted. The protocol still functions properly as shown by our proof (e.g., this scenario does not produce unnecessary retransmissions), but it is a peculiar feature. It arises because unsolicited status messages do not contain the full status of the receiver.

4. POTENTIAL PROBLEMS

In the previous section, we examined the proposed ATM retransmission scheme under normal operating conditions. Here, we consider two of the problems that can arise if some of the conditions enumerated in Section 2 do not hold. For a more detailed description of potential problems, see [7].

4.1 Out-of-Sequence Traffic

There are many scenarios where frames that arrive at the destination out-of-sequence result in unnecessary retransmissions. For example, assume copy 1 of frame X+1 is transmitted after copy 1 of frame X, but arrives at the destination before it. An unsolicited STAT will be sent NACKing frame X, and frame X will be unnecessarily retransmitted (see Figure 4).

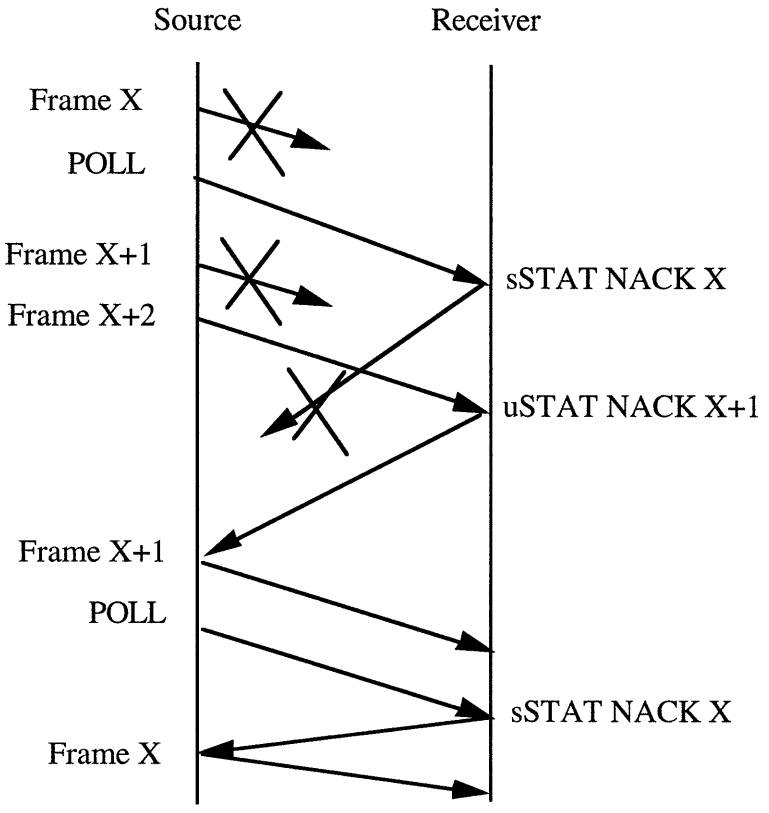

Figure 3 An unsolicited STAT NACKing frame X+1 arrives at the source before a solicited STAT NACKing frame X. Thus the second copy of frame X+1 is sent before the second copy of frame X.

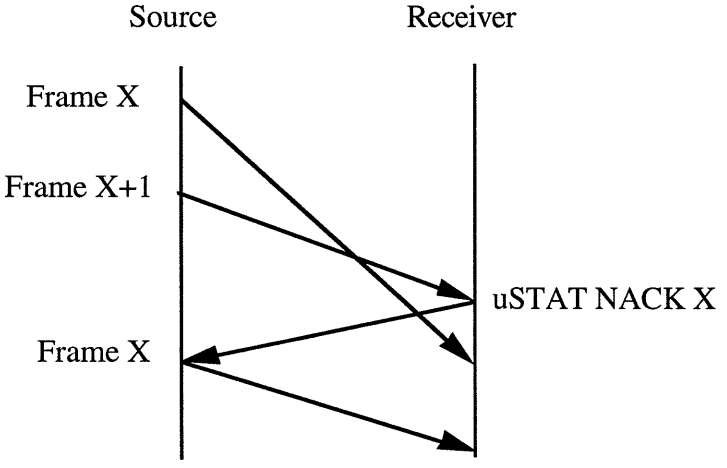

Figure 4 Frame X+1 travels out of sequence, causing frame X to be unnecessarily retransmitted.

Unnecessary retransmissions may lead to serious failures of the protocol. In Section 3, where we proved that the retransmission protocol works properly, it was assumed that unnecessary retransmissions do not occur. With this assumption, the destination was guaranteed not to receive frames prior to SEQR, since SEQR indicates the destination has

received all frames through SEQR-1. Without this assumption, the protocol will likely fail, as is demonstrated in the following example.

Assume SEQA equals 0 and SEQ equals 100. Assume frames 0 through 50 have been received by the destination. Thus, SEQR and SEQH equal 51. Next, assume frame number 5 is retransmitted unnecessarily. This extra copy of frame 5 will be stored in the resequencing buffer at the destination, and will be treated as frame 5 in the 'next cycle' of 2^{24} frames. Thus, an incorrect frame will be passed up to the higher layer at the destination. Also, when frame 5 arrives at the destination, the receiver will interpret this as being a frame 'greater than' SEQH. Thus, it will send an unsolicited STAT NACKing frames 'between' 51 and 4, inclusive, and it will set SEQH to 5. This erroneous NACK could result in more unnecessary retransmissions (although if the source realizes that the status message does not make sense, it would ignore it). The value of SEQH will be incorrect so that the unsolicited STAT mechanism will be corrupt.

4.2 Consecutive Lost Polls

Condition 7 in Section 2 stated that no more than 2^{24}-2 consecutive poll/status message combinations are lost. If this condition does not hold, and the source adheres to condition 9, then the source would be unable to send any more polls. The source may be able to partially rely on unsolicited STATs for NACKs. However, if the last frames of a connection are lost, or if an unsolicited STAT is lost, then some frames will never be retransmitted. Or, if 2^{24}-1 consecutive frames are successfully transmitted, the destination will not be able to ACK these frames due to the lack of polls. Due to the restriction of condition 8, the source will be unable to transmit any more frames. The protocol will be deadlocked.

To deal with this situation, the proposed scheme includes a variable Timer_NO-RESPONSE. If this timer expires without the source receiving a solicited STAT, the connection is terminated.

5. CONCLUSIONS

We have proved that under reasonable conditions, the proposed ATM retransmission scheme will generate retransmissions of lost frames without producing unnecessary retransmissions. It satisfies the conditions of both safety and liveness. The operation of the protocol relies on the fact that data is expected to travel in sequence in ATM; if frames do travel out-of-sequence, the retransmission protocol will likely fail.

References

[1] CCITT Study Group XI Document DT/11/3-28, "Service specific connection oriented (SSCOP) specification," May 23, 1993.

[2] CCITT Recommendation I.363, "B-ISDN ATM Adaptation Layer (AAL) Specification, Geneva, 1992.

[3] Simmons, J., "Performance analysis of poll-based retransmission schemes", submitted to *IEEE/ACM Transactions on Networking*, 1993.

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

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

[6] Bertsekas, D. and Gallager, R., *Data Networks*, Second Edition, Englewood Cliffs, NJ: Prentice-Hall, Inc., 1992.

[7] Simmons, J., *End-to-End Reliable Communication in Data Networks*, PhD Thesis, MIT, August, 1993.