

# Proof of correctness of ATM retransmission scheme <sup>1</sup>

Jane M. Simmons <sup>2</sup>

*AT & T Labs, Room 1L-231, 101 Crawfords Corner Rd, Holmdel, NJ 07733-3030, USA*

Accepted 29 November 1995

---

## Abstract

The ITU has designed a poll-based retransmission scheme as part of the Service Specific Connection Oriented Protocol for ATM systems. The basic scheme consists of the transmitter periodically sending polls to the destination indicating which frames have been sent, and the receiver 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 operates correctly. In this paper, it is shown that the scheme does satisfy the properties of liveness and safety: the source can continue forever to accept frames for transmission, and all frames are eventually delivered in proper sequence at the destination, assuming that certain conditions hold. In addition, it is shown that unnecessary retransmissions do not occur.

*Keywords:* Asynchronous Transfer Mode (ATM); B-ISDN; Retransmission schemes; ATM Adaptation Layer services; Service Specific Connection Oriented Protocol (SSCOP); Protocol verification

---

## 1. Introduction

Asynchronous Transfer Mode (ATM) is a network protocol currently being designed for use on Broadband Integrated Services Digital Network (B-ISDN) systems. ATM provides a common format for transmitting voice, data, and video over B-ISDN systems. The two major layers comprising the ATM protocol are the ATM Adaptation Layer (AAL) and the ATM layer (see Fig. 1). At the transmitter, the AAL receives data from a higher layer, packages the data into frames, and divides the frames into fixed

length segments. The segments are passed down to the ATM layer, which adds a fixed amount of control information to each segment to form what is called a cell. The ATM layer is responsible for cell routing. (In general, we will use the term “ATM” to refer to ATM systems and not the specific ATM layer.) At the receiver, the AAL is responsible for reconstructing the frame, checking the integrity of the frame, and requesting retransmission of an errored frame, if necessary.

The AAL is comprised of two sublayers: the lower sublayer is referred to as the AAL Common Part and the higher sublayer is referred to as the Service Specific Convergence Sublayer (SSCS) [5]. Segmentation and reconstruction of frames, as well as frame integrity checks, occur in the Common Part. Currently, five Common Part protocols have been

---

<sup>1</sup> This work was performed at the Laboratory for Information and Decision Systems, MIT, Cambridge, MA. It was funded by the NSF and ARPA.

<sup>2</sup> Email: jsimmons@lids.mit.edu.

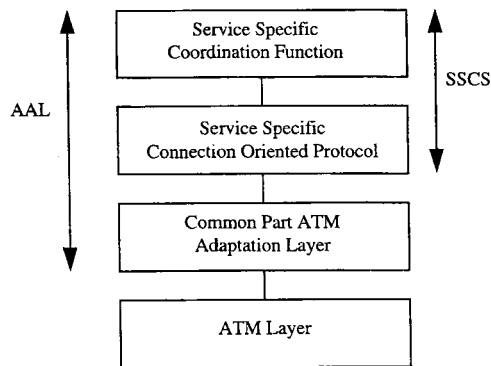


Fig. 1. ATM and AAL structure.

defined; they are referred to as AAL Common Part Type 1 through Type 5, where each Common Part protocol corresponds to a specific class of service provided by ATM [4]. For example, AAL Common Part Type 5 is designed for connection oriented, variable bit rate traffic, with no required timing between transmitter and receiver.

The SSCS is itself comprised of two sublayers: the Service Specific Coordination Function (SSCF) and the Service Specific Connection Oriented Protocol (SSCOP). The main functions of the SSCOP are to provide mechanisms for the establishment and release of connections, and to provide assured data transfer and flow control between AAL peer entities. A single SSCOP has been defined that can operate over different Common Part protocols [5]. The SSCF maps the signalling requirements of the protocol layer above the AAL, referred to as layer 3, to the SSCOP services [6,7]. For example, the SSCF-UNI operates at the User-to-Network Interface to coordinate the services required by the layer 3 user and the services provided by the SSCOP, such as "Connection Establish". Similarly, the SSCF-NNI operates at the Network Node Interface.

This paper focuses on the assured data transfer capability provided by the SSCOP. The SSCOP is responsible for ensuring that any lost or corrupt frames are retransmitted (corrupt frames are dropped by the Common Part protocol). The retransmission scheme defined in [5] is based on a polling mechanism, where the transmitter periodically sends poll messages to the receiver indicating which frames have been sent, and the receiver responds with a

status message indicating which of these frames have not been received. In addition, 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 the frame with sequence number  $N$  without having received the frame with sequence number  $N - 1$ , it immediately sends a Negative Acknowledgement (NACK) of frame  $N - 1$ , rather than waiting for a poll.

A large portion of the retransmission protocol is in place to ensure that the scheme does not produce any unnecessary retransmissions. With this added complexity, it is not readily apparent whether the scheme generates the necessary retransmissions. This paper presents a high level proof that the retransmission scheme works correctly, assuming some corrections are made to the specification in [5]. It is shown that the source AAL entity 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 AAL entity, assuming that certain conditions hold. In addition it is shown that under these conditions, unnecessary retransmissions do not occur.

Section 2 provides the details of the retransmission scheme, and transforms the definition of [5] into algorithmic form. In Section 3.1, the conditions that must hold in order for the protocol to work properly are defined. In Sections 3.2-3.4, the protocol is shown to work correctly, assuming all sequence numbers are integers that can increase without bound. In Section 3.5, the proof is extended to show that proper operation is maintained when sequence numbers are integers modulo  $2^{24}$ , as specified in [5].

## 2. Description of retransmission scheme

First, we provide an overview of the retransmission scheme defined in [5]. In Section 2.2, we transform the protocol defined in [5] into algorithmic form. An example is provided in Section 2.3 that illustrates many of the details of the protocol. Throughout the analysis, the variable names defined in [5] are used; in general, the notation  $VT()$  indicates a variable maintained at the transmitter, and

Information		
PAD		
Type	SD Sequence Number N(S)	

Fig. 2. Sequenced data PDU.

VR() indicates a variable maintained at the receiver. The terms transmitter and receiver refer to the AAL peer entities that are the source and destination of the data, respectively.

2.1. Overview of ATM retransmission scheme

There are four types of SSCOP Protocol Data Units (PDUs) that are of concern in the retransmission scheme; they are shown in Figs. 2-5. (A PDU is the data unit exchange between peer entities.) Sequenced Data (SD) PDUs carry the actual data from transmitter to receiver; SDs are numbered sequentially modulo  $2^{24}$ . In the Introduction, we informally referred to SD PDUs as frames. After an SD is transmitted, it is stored in the transmission buffer, which is assumed to be large enough to hold all SDs that have been transmitted but not acknowledged by the receiver.

If the SD is received without errors at the receiver, and the SD sequence number falls within the receive window, then the SD is placed in the receiver buffer. It is assumed that the receiver buffer is large enough to hold all SDs falling within the receive window. An SD is delivered to the higher layer at the receiver only when all SDs with smaller sequence numbers have been received.

POLL PDUs are periodically sent by the transmitter to indicate the status of the transmitter and to query the status of the receiver. Each POLL is numbered sequentially modulo  $2^{24}$ , independently of the SDs.

Whenever the receiver receives a POLL, it responds with a solicited status message PDU, referred

Poll Sequence Number N(PS)	
Type	Next SD Sequence Number to be sent N(S)

Fig. 3. POLL PDU.

PAD	NACK List Element 1
PAD	NACK List Element 2
	⋮
PAD	NACK List Element L
Reserved	Poll Sequence Number N(PS)
Reserved	Receive Window N(MR)
Type	Next Expected SD Number N(R)

Fig. 4. STAT PDU.

to as a STAT. The STAT indicates *all* SDs that have not been received. As shown in Fig. 4, the NACK list format is L1, L2, L3, L4, ..., which indicates that SDs L1 through L2-1 are missing, SDs L3 through L4-1 are missing, etc. ( $L_i$  is the  $i$ th NACK list element). For example, if the receiver has received SDs 1, 2, 5, 6, and 8, its NACK list would include "3, 5, 7, 8" to indicate SDs 3, 4, and 7 are missing. The STAT also includes the sequence number of the incoming POLL.

In addition to sending STATs in response to POLLS, the receiver sends *unsolicited* status messages, called USTATs, whenever it receives an SD with sequence number  $N$  without having received an SD with sequence number  $N - 1$ . The receiver takes advantage of the fact that SDs are sequentially numbered and are expected to arrive at the receiver in the same order in which they were sent by the transmitter, to send an immediate NACK rather than waiting for a POLL. For example, if SDs 1, 2, and 5 arrive at the receiver in that order, then as soon as SD 5 is received, the receiver will send a USTAT to the transmitter NACKing SDs 3 and 4. The one restriction on USTATs is that *a USTAT may not NACK an SD that has been previously NACKed in another status message (either STAT or USTAT)*. Thus, a

PAD	NACK List Element 1
PAD	NACK List Element 2
Reserved	Receive Window N(MR)
Type	Next Expected SD Number N(R)

Fig. 5. USTAT PDU.

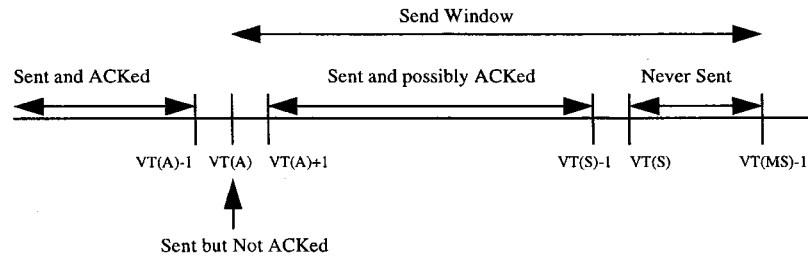


Fig. 6. Variables relating to SD sequence numbers that are maintained by the transmitter.

USTAT does not necessarily include the full state information of the receiver. The rationale for this restriction is explained below.

The retransmission process at the transmitter has added complexity in order to prevent unnecessary retransmissions. The transmitter maintains the variable  $VT(PS)$  to indicate the sequence number of the last POLL transmitted. After an SD is transmitted, it is stored in the transmission buffer and "stamped" with the current value of  $VT(PS)$ . When the source receives a STAT, it retransmits only those NACKed SDs that are stamped with a  $VT(PS)$  value that is less than the poll sequence number indicated in the STAT. Essentially the retransmission criterion is: retransmit any NACKed SDs that were sent before the POLL that triggered the NACK. As will be shown in Section 3.4, this ensures that the retransmission must be necessary.

When the transmitter receives a USTAT, it retransmits all NACKed SDs, regardless of the value of  $VT(PS)$  stored with the SD in the transmission buffer. A comparison of  $VT(PS)$  does not have to be performed since USTATs only NACK SDs that have

not previously been NACKed. Eliminating the comparison with  $VT(PS)$  was the rationale for the ITU specifying that the USTAT carry only partial state information.

The source also uses STATs and USTATs to release ACKed SDs from the transmission buffer. There are two options for releasing SDs: release all ACKed SDs, or, release an ACKed SD only if all lower-numbered SDs have been ACKed. The source chooses which method to use at the time the connection is set up. For the remainder of the paper, we will assume the latter option.

As implied by the above description, selective retransmission, as opposed to Go Back  $N$ , is employed. For a discussion of general Go Back  $N$  and selective repeat systems, see [1,8].

## 2.2. Transmitter and receiver algorithms

In the previous section, an overview of the retransmission scheme was provided. In this section, the specific variables that must be maintained by the transmitter and receiver are presented, along with an

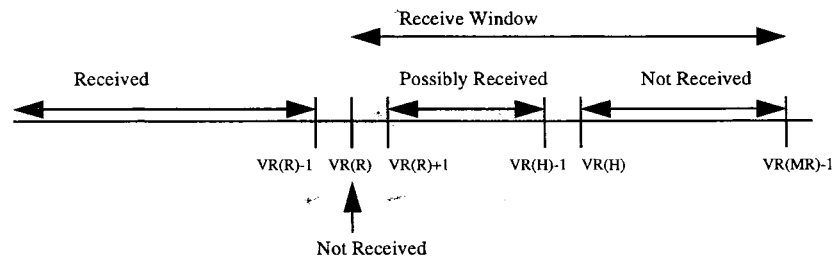


Fig. 7. Variables relating to SD sequence numbers that are maintained by the receiver.

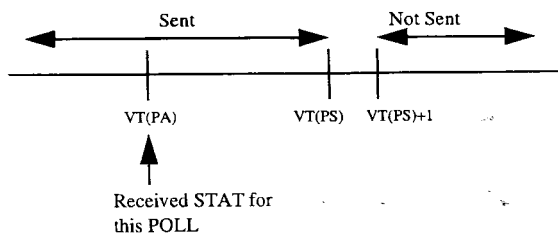


Fig. 8. Variables relating to POLL sequence numbers that are maintained by the transmitter. POLL sequence numbers are independent of SD sequence numbers.

algorithmic statement of the protocol. See Figs. 6 through 8 for the relationship among the different variables. For simplicity, all variables and sequence numbers in this section are treated as ordinary integers that can increase without bound, rather than integers modulo  $2^{24}$ ; thus, the notions of “less than” and “greater than” are unambiguous. The terminology SD  $X$  is used to indicate the SD with sequence number  $X$ .

#### Transmitter variables.

- $VT(S)$  = Sequence number of next SD to be sent for the first time. SDs 1 through  $VT(S) - 1$  have been sent at least once. SDs  $\geq VT(S)$  have never been sent.
- $VT(A)$  = Oldest transmitted but unacknowledged frame at transmitter. SDs 1 through  $VT(A) - 1$  have been ACKed. SD  $VT(A)$  has not been ACKed.
- $VT(MS)$  = One greater than the upper edge of the SD send window. SDs  $VT(A)$  through  $VT(MS) - 1$  can be transmitted. SDs  $\geq VT(MS)$  cannot be transmitted.
- $VT(PS)$  = Sequence number of poll most recently transmitted. POLLS 1 through  $VT(PS)$  have been transmitted. POLLS  $> VT(PS)$  have never been transmitted.
- $VT(PA)$  = The POLL sequence number contained in the STAT most recently received by transmitter. A STAT in response to POLL  $VT(PA)$  has been received by the transmitter. STATs in response to POLLS  $> VT(PA)$  have not been received by the transmitter.
- $SDX.VT(PS) = VT(PS)$  at the time SD  $X$  was most recently transmitted.

**Receiver variables.** (Below, the term received is used to imply “received without errors”.)

- $VR(R)$  = Sequence number of lowest numbered SD that the receiver has not received. SDs 1 through  $VR(R) - 1$  have been received. SD  $VR(R)$  has not been received.
- $VR(H)$  = One greater than the highest numbered SD that the receiver knows the transmitter has sent. SD  $VR(H) - 1$  has been received, or a POLL sent after SD  $VR(H) - 1$  was transmitted has been received. SDs  $\geq VR(H)$  have not been received. POLLS sent after SD  $VR(H)$  was transmitted have not been received.
- $VR(MR)$  = One greater than the upper edge of the SD receive window. SDs  $VR(R)$  through  $VR(MR) - 1$  can be accepted at the receiver. SDs  $\geq VR(MR)$  will be dropped.

#### PDU fields.

- $SD.N(S)$  = value of  $VT(S)$  at the time the SD is generated.
- $POLL.N(S)$  = value of  $VT(S)$  at the time the POLL is generated.
- $POLL.N(PS)$  = value of  $VT(PS)$  at the time the POLL is generated.
- $STAT.N(PS)$  = value of  $VT(PS)$  contained in the POLL that generated the STAT.
- $STAT.N(R)$  = value of  $VR(R)$  at the time the STAT is generated.
- $USTAT.N(R)$  = value of  $VR(R)$  at the time the USTAT is generated.
- $STAT.N(MR)$  = value of  $VR(MR)$  at the time the STAT is generated.
- $USTAT.N(MR)$  = value of  $VR(MR)$  at the time the USTAT is generated.

#### Algorithm at Transmitter

##### Initialization:

Set  $VT(S) = VT(A) = VT(PS) = VT(PA) = 0$ .  
 $VT(MS) = W$ , where  $W$  is specified when the connection is established.  $W$  must be less than  $2^{24}$ .

**SD original transmission:** (perform within finite intervals chosen by the source)

```
if ((SD on transmission queue) && (VT(S) < VT(MS))) {
```

```

Set  $SD.N(S) = VT(S)$ 
Transmit  $SD$ 
Remove  $SD$  from transmission queue and store
 $SD$  in transmission buffer
Set  $SD.VT(PS) = VT(PS)$  /*  $SD$  is stored in
transmission buffer with current value of
 $VT(PS)$  */
Set  $VT(S) = VT(S) + 1$ 
}

```

**SD retransmission:** (perform whenever an  $SD$  is on retransmission queue)

(retransmissions are given priority over original transmissions)

```

Transmit  $SD$ 
Remove  $SD$  from retransmission queue
Set  $SD.VT(PS) = VT(PS)$ 

```

**POLL transmission:** (perform within finite intervals chosen by the source)

```

if ( $VT(PS) < VT(PA) + 2^{24} - 1$ ) {
  Set  $VT(PS) = VT(PS) + 1$ 
  Set  $POLL.N(PS) = VT(PS)$ 
  Set  $POLL.N(S) = VT(S)$ 
  Transmit  $POLL$ 
}

```

**STAT reception:** (perform whenever a  $STAT$  is received by transmitter)

```

if ( $STAT.N(R) \geq VT(A)$ ) { /* this should always
be the case for any arriving  $STAT$  */
  Remove all  $SD X$  from transmission buffer where
   $VT(A) \leq X < STAT.N(R)$ 
  Set  $VT(A) = STAT.N(R)$ 
  Set  $VT(MS) = STAT.N(MR)$ 
  for each NACKed  $SD X$  indicated in  $STAT$  {
    if ( $(SDX.VT(PS) < STAT.N(PS))$  &&
    ( $SD X$  not already in retransmission queue)) {
      Retrieve  $SD X$  from transmission buffer
      and place in retransmission queue
    }
  }
  Set  $VT(PA) = STAT.N(PS)$ 
}

```

**USTAT reception:** (perform whenever a  $USTAT$  is received by transmitter)

```

if ( $USTAT.N(R) \geq VT(A)$ ) { /* this should al-
ways be the case for any arriving  $USTAT$  */

```

```

Remove all  $SD X$  from transmission buffer where
 $VT(A) \leq X < USTAT.N(R)$ 
Set  $VT(A) = USTAT.N(R)$ 
Set  $VT(MS) = USTAT.N(MR)$ 
for each NACKed  $SD X$  indicated in  $USTAT$  {
  Retrieve  $SD X$  from transmission buffer and
  place in retransmission queue
}
}

```

### Algorithm at Receiver

#### Initialization:

```

Set  $VR(R) = VR(H) = 0$ ;  $VR(MR) = W$  (same  $W$  as
for transmitter)

```

#### SD reception:

```

if ( $(SD.N(S) = VR(R))$  && ( $SD.N(S) < VR(MR)$ ))
{
  Pass  $SD$  to higher layer
   $VR(R) = VR(R) + 1$ 
  while ( $VR(R) = SD.N(S)$  of an  $SD$  in receiver
  buffer) {
    Remove  $SD$  from receiver buffer and pass  $SD$ 
    to higher layer
    Set  $VR(R) = VR(R) + 1$ 
  }
  if ( $SD.N(S) = VR(H)$ ) {
    Set  $VR(H) = SD.N(S) + 1$ 
  }
}
else if ( $VR(R) < SD.N(S) < VR(MR)$ )) {
  Store  $SD$  in receiver buffer
  if ( $SD.N(S) > VR(H)$ ) {
    Generate USTAT NACKing all  $SD$ s between
     $VR(H)$  and  $SD.N(S) - 1$ , inclusive
    Set  $USTAT.N(R) = VR(R)$ 
    Set  $USTAT.N(MR) = VR(MR)$ 
    Set  $VR(H) = SD.N(S) + 1$ 
  }
  if ( $SD.N(S) = VR(H)$ ) {
    Set  $VR(H) = SD.N(S) + 1$ 
  }
}
else if ( $SD.N(S) \geq VR(MR)$ ) {
  Drop  $SD$ 
  if ( $VR(MR) > VR(H)$ ) { /* implies  $SD.N(S)$ 
  >  $VR(H)$  */
    Generate USTAT NACKing all  $SD$ s between
     $VR(H)$  and  $VR(MR) - 1$ , inclusive
  }
}

```

```

Set USTAT.N(R) = VR(R)
Set USTAT.N(MR) = VR(MR)
Set VR(H) = VR(MR)
}
}

```

**POLL reception:**

```

if (POLL.N(S) ≥ VR(H)) { /* this should be the
case for any arriving POLL */
Set VR(H) = min(POLL.N(S), VR(MR))
Generate STAT NACKing all SDs between VR(R)
and min (POLL.N(S) - 1, VR(MR) - 1) inclu-
sive, that are not in receiver buffer.
Set STAT.N(R) = VR(R)
Set STAT.N(MR) = VR(MR)
Set STAT.N(PS) = POLL.N(PS)
}
}

```

Scenarios not included in the algorithm above are considered errors in the protocol. For example, receiving an SD with a sequence number less than  $VR(R)$  is an error condition, and triggers an error recovery procedure. Recovering from errors in the protocol is not discussed in this paper.

The protocol definition in [5] is inconsistent with respect to the definition of  $VT(PA)$ . In one section of [5],  $VT(PA)$  is set equal to the POLL sequence number contained in the most recently received STAT; in another section,  $VT(PA)$  is treated as being one greater than the POLL sequence number contained in the most recently received STAT. The protocol could work with either definition of  $VT(PA)$ , as long as it is consistent throughout. We have chosen to treat  $VT(PA)$  as being equal to the POLL sequence number contained in the most recently received STAT. Thus, in the initialization step at the transmitter, we set  $VT(PA)$  to 0; in [5], it is initialized to 1. In Section 3.5.3, another problem with the manner in which  $VT(PA)$  is updated is discussed.

Setting the value of the receive window,  $VR(MR)$ , is not precisely defined in the protocol. The receiver is permitted to decrease  $VR(MR)$ , however,  $VR(MR)$  must always be greater than or equal to  $VR(H)$ .

Throughout our discussion, it is assumed that each of the functional blocks in the above algorithm can be viewed as an indivisible operation i.e., once a given block of operations is started, no other opera-

tions are performed until the entire block is finished. When we refer to transmissions at the transmitter and receiver, we assume the PDU 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 Fig. 9. The important points are described below.

In the example, SDs 1 and 2 are delivered successfully. SD 3 is lost, and POLL 1 generates a STAT that NACKs SD 3. Note that the arrival of the POLL also causes  $VR(H)$  to be updated to 4 (i.e., due to the POLL, the destination knows that all SDs through 3 have been sent at least once). SD 4 is lost; SD 5 is received successfully and triggers a USTAT that NACKs SD 4. The USTAT does not NACK SD 3 since that was NACKed in a previous STAT.

The  $VT(PS)$  value stored with SD 3 is 0; thus, STAT 1 triggers the retransmission of SD 3 (since  $0 < 1$ ). The USTAT automatically triggers the retransmission of SD 4 without any  $VT(PS)$  comparisons being necessary.

Before the retransmissions of SDs 3 and 4 are sent, POLL 2 is sent, which generates a STAT that NACKs both SDs 3 and 4. STAT 2 triggers no retransmissions since the  $VT(PS)$  value stored with SDs 3 and 4 is now 2. The retransmission of SD 3 is lost, but the retransmission of SD 4 is successful.

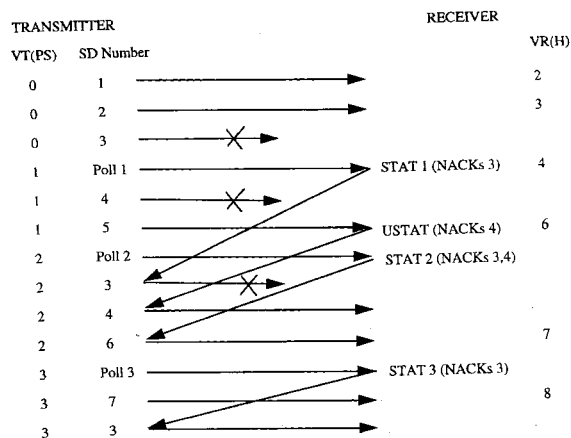


Fig. 9. Operation of ATM retransmission scheme.

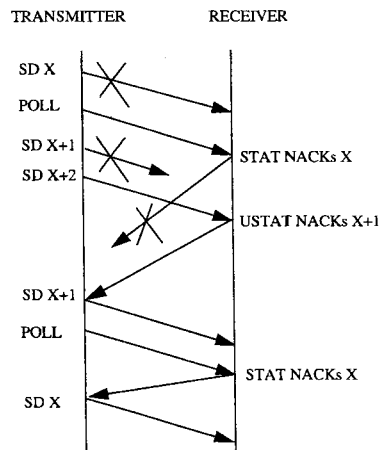


Fig. 10. A USTAT NACKing SD  $X+1$  arrives at the source before a STAT NACKing SD  $X$ . Thus the retransmission of SD  $X+1$  is sent before the retransmission of SD  $X$ .

Even though the destination receives SD 4 and not SD 3, it does not send a USTAT, since  $VR(H)$  is 6. Finally, POLL 3 generates STAT 3 that NACKs SD 3. This triggers the successful retransmission of SD 3.

It is interesting to note one peculiarity which arises from the fact that USTATs do not contain the full status of the receiver. Consider the following example, which is depicted in Fig. 10. Assume the destination sends a STAT NACKing SD  $X$ , and later sends a USTAT NACKing SD  $X+1$ . The USTAT will not NACK SD  $X$  since USTATs can only NACK SDs that have not been NACKed previously. Assume the STAT NACKing SD  $X$  is lost. When the USTAT arrives, SD  $X+1$  will be retransmitted before SD  $X$  is retransmitted. Thus, retransmissions can occur “out of sequence,” although the protocol still functions properly as shown in the next several sections.

### 3. Proof of proper operation

The goal of this section is to provide a high level proof of the correctness of the protocol which was described algorithmically above. We must show that the transmitter can continue forever to accept SDs for transmission from the higher layer, and that all SDs are eventually delivered in proper sequence at

the receiver. We also will show that the property of no unnecessary retransmissions holds.

#### 3.1. Conditions of normal operation

It is easy to come up with special circumstances where the ATM retransmission scheme does not work correctly. Thus, we first must define the conditions under which we can prove the correctness of the protocol. These conditions are:

- (1) In the initial state of the system, there are no PDUs on any of the links.
- (2) A PDU cannot be received before it is sent.
- (3) Undetected errors do not occur.
- (4) State information at the transmitter and receiver is not lost.
- (5) PDUs travel in order on the links, i.e., if PDU  $A$  is transmitted before PDU  $B$ , then if both PDUs are not lost, PDU  $A$  will always arrive at the receiver before PDU  $B$ .
- (6) The connection does not go down.
- (7) Any SD which is repeatedly transmitted will eventually be received.
- (8) All transmission and propagation delays are finite.
- (9) No more than  $2^{24} - 2$  consecutive POLL/STAT combinations are lost (i.e., one out of every  $2^{24} - 1$  consecutive POLLS arrives successfully at the receiver, and the corresponding STAT arrives successfully at the transmitter).

Below, we provide a high level proof that the protocol satisfies the properties of both *safety* and *liveness*, given the above conditions. In general, safety is the property that any actions performed by the protocol are in accord with the services required of the protocol, and liveness is the property that the required services will be provided in finite time. (For a discussion of general protocol verification methodologies, see [2].) Conditions (1) through (4) above are needed to prove that the protocol is safe, and conditions (6) through (9) are needed to show that the protocol is live. Condition (5) is needed to show that unnecessary retransmissions do not occur and to show that the protocol continues to work properly when sequence numbers are maintained modulo  $2^{24}$ .

In Sections 3.2–3.4, safety and liveness and the property of no unnecessary retransmissions are shown

to be satisfied if all sequence numbers are treated as integers that can increase without bound. In Section 3.5, the proof is extended to show that the protocol remains correct when SD sequence numbers and POLL sequence numbers are integers modulo  $2^{24}$ . Some of the techniques for the proof are from [1] and [3].

### 3.2. Safety condition

The higher layer at the transmitter is a source of sequentially numbered SDs. Let  $\mathcal{X}$  represent the sequence of SDs that have been generated by the source, i.e.,  $\mathcal{X} = \{SD1, SD2, \dots, SDN\}$ . At the receiver, the higher layer acts as a “sink” of SDs. Let  $\mathcal{Y}$  represent the sequence of SDs that have been delivered to the sink. The safety condition for the system is that  $\mathcal{Y}$  must be an initial sequence of  $\mathcal{X}$ , i.e.,  $\mathcal{Y} = \{SD1, SD2, \dots, SDM\}$  for  $M \leq N$ .

From the protocol definition for delivering SDs to the higher layer (see SD Reception in the algorithm above), SDs are always delivered in sequential order. Combining this with conditions (1), (2), and (3) above (i.e., “nothing comes out that was not put in”, and there can be no undetected errors in the sequence number), we see that the safety condition must always be satisfied.

### 3.3. Liveness condition

The algorithm is live if the transmitter can continue forever to accept data from the higher layer, and the receiver continues to deliver it to the higher layer in finite time.

The liveness proof presented below relies on the assumption that within a finite time, some POLL/STAT combination will be successful (see condition (9) in Section 3.1), i.e., a POLL arrives at the receiver error-free, and the corresponding STAT arrives at the transmitter error-free. Thus, there is a periodic exchange of state variables between the transmitter and receiver. To show liveness, we only need to consider STATs; USTATs are not necessary for liveness.

Consider the value of  $VR(R)$  at any given time, and assume it equals  $V$ . ( $VR(R)$  is the next consecutive SD to be received.) We show that the transmitter

is able to transmit SD  $V$ , SD  $V$  will eventually be received at the destination, and  $VR(R)$  will be incremented.

The receive window extends from  $VR(R)$  to  $VR(MR) - 1$ . Since the receive window is assumed to be at least 1, we know that  $VR(MR) > VR(R)$ . (If the receive window were never greater than zero, we would not expect the algorithm to be live.) Due to the periodic exchange of state variables in POLLS and STATs,  $VT(MS)$  is eventually  $> VR(R)$  (since  $VT(MS)$  is set equal to the value of  $VR(MR)$  contained in a STAT). Thus, the send window will allow the source to send SD  $V$ .

Let  $P$  equal the value of  $VT(PS)$  at the time SD  $V$  is transmitted. ( $VT(PS)$  is the sequence number of the POLL that was most recently sent by the transmitter.) Thus, SD  $V$  will be “stamped” with  $P$  when it is stored in the transmission buffer; the stamp cannot be changed unless SD  $V$  is retransmitted. We know that some POLL transmitted after SD  $V$  is transmitted will be successful.  $VT(PS)$  is incremented immediately before a POLL is sent; thus, for the successful POLL,  $POLL.N(PS) > P$ . Since  $POLL.N(S)$  indicates the lowest numbered SD that has not been transmitted, it must also be true that  $POLL.N(S) > V$ .

When the POLL arrives at the receiver, either SD  $V$  will have already arrived or SD  $V$  is lost, since PDUs travel in sequence. If SD  $V$  is lost, the generated STAT will NACK SD  $V$  since  $V$  is less than both  $POLL.N(S)$  and  $VR(MR)$ . The STAT will contain  $STAT.N(PS) = POLL.N(PS)$ . If SD  $V$  is NACKed in the STAT, then at the transmitter, the STAT will generate the retransmission of SD  $V$  since  $STAT.N(PS) > P$ . (SD  $V$  should still be in the transmission buffer since an ACK of SD  $V$  should not have been received.) Thus, if SD  $V$  is lost, it will be retransmitted.

The arguments in the previous paragraph hold regardless of whether the transmission is the original transmission or a retransmission of SD  $V$ . Thus, due to condition (7), SD  $V$  will eventually be accepted at the receiver, and due to condition (8), this will occur in finite time. Since  $V$  equals  $VR(R)$ , SD  $V$  will be delivered to the higher layer,  $VR(R)$  will be incremented, and, if necessary,  $VR(MR)$  will be incremented to ensure that  $VR(MR) > VR(R)$ .

The algorithm has been shown to be live. Next,

we will show that the algorithm does not produce any unnecessary retransmissions.

### 3.4. No unnecessary retransmissions

We define an unnecessary retransmission as occurring when copy  $i + 1$  of an SD is sent even though copy  $i$  is not lost. (Copy 1 is the original transmission, copy 2 is the first retransmission, etc.) In the discussion below, we examine whether copy  $i + 1$  of an arbitrary SD, say SD  $X$ , could ever be unnecessarily sent. In order for copy  $i + 1$  to be unnecessary, it must be true that copy  $i$  of SD  $X$  does arrive error-free at the destination. (For simplicity, we will use the term “arrive” to imply “arrive error-free”.)

There are two possible ways a NACK of SD  $X$  can be generated:

- (1) SD  $Y$  arrives at the destination, and at the time of its arrival SD  $X$  has not arrived, and one of the following holds:  $VR(MR) > Y > X \geq VR(H)$  or  $Y \geq VR(MR) > X \geq VR(H)$ . A USTAT NACKing SD  $X$  is sent and  $VR(H)$  is updated to  $\min(Y + 1, VR(MR))$ .
- (2) A POLL arrives at the destination, and at the time of its arrival SD  $X$  has not arrived, and the following holds:  $POLL.N(S) > X$  and  $VR(MR) > X$ . A STAT NACKing SD  $X$  is sent and  $VR(H)$  is updated to  $\min(POLL.N(S), VR(MR))$ .

First, consider the case where  $i = 1$ . Assume copy 1 of SD  $X$  arrives at the destination. Copy 1 of SD  $X$  must be sent before any copy of SD  $Y$ , where  $Y > X$ , and must be sent before a POLL with  $POLL.N(S) > X$ . Given that PDUs travel in order, such an SD  $Y$  or such a POLL cannot arrive before SD  $X$ . Thus, the conditions in the two procedures above are not satisfied; copy 1 of SD  $X$  will not be NACKed, so no further copies of SD  $X$  will be sent.

Next, consider the case where  $i > 1$ . Since SD  $X$  has been sent more than once, and since SDs are only retransmitted in response to NACKs, it must be true that SD  $X$  was NACKed at least once. From the procedures described in statements 1 and 2 above, it must be true that after SD  $X$  is NACKed,  $VR(H)$  is updated to a value greater than  $X$ .  $VR(H)$  is nondecreasing. Thus, after SD  $X$  has been NACKed once, statement 1 above can never be satisfied, since it

requires  $X \geq VR(H)$  (this is the mechanism that ensures a USTAT can only NACK copy 1 of an SD).

Thus, we only need to consider statement 2 above. Let  $t_i$  be the time that copy  $i$  of SD  $X$  is transmitted and let  $SDX.VT(PS)$  be the poll stamp associated with copy  $i$  of SD  $X$  (i.e.,  $SDX.VT(PS)$  equals the value of  $VT(PS)$  at time  $t_i$ ). 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 SD  $X$ .

First, consider a POLL sent before  $t_i$ . Its poll sequence number satisfies  $POLL.N(PS) \leq SDX.VT(PS)$ , and the corresponding STAT contains  $STAT.N(PS) = POLL.N(PS)$ . Such a STAT cannot result in copy  $i + 1$  of SD  $X$  being transmitted since  $STAT.N(PS)$  is not greater than  $SDX.VT(PS)$ . Next, consider a POLL sent after  $t_i$ . If copy  $i$  of SD  $X$  arrives at the destination, then it must arrive before such a POLL, since it is assumed PDUs travel in sequence. Thus, the POLL would not NACK SD  $X$ .

Thus, a POLL will not trigger an unnecessary retransmission of SD  $X$ . Overall, it has been shown that given copy  $i$  of SD  $X$  is received, copy  $i + 1$  will not be sent. Thus, no unnecessary retransmissions are produced by the protocol.

### 3.5. Sequence numbers with modulus

In the sections above, it was shown that the retransmission protocol works correctly if sequence numbers can increase without bound. In this section, we show that the protocol continues to work if SD sequence numbers and POLL sequence numbers are treated modulo  $2^{24}$ . The proof heavily relies on the condition that PDUs travel in sequence on links.

First, we need to define “greater than” and “less than” when dealing with integers modulo  $2^{24}$ . One can envision the numbers from 0 to  $2^{24} - 1$  on a circle, increasing clockwise. At the transmitter, we assume the “base” for SD sequence numbers is  $VT(A)$ . Thus, for any number  $Z$  between 0 and  $2^{24} - 1$ , “less than  $Z$ ” is interpreted as any number that falls in the region that extends clockwise from  $VT(A)$  to  $(Z - 1) \bmod 2^{24}$ , and “greater than  $Z$ ” is interpreted as any number that falls in the range from  $(Z + 1) \bmod 2^{24}$  clockwise to  $(VT(A) - 1) \bmod 2^{24}$ . The sequence number “base” at the receiver is

$VR(R)$ ; the POLL sequence number “base” at the transmitter is  $VT(PA)$ .

We must check that the acceptance policy at the receiver, the NACK procedure, and the retransmission procedure are unaffected if the modulus is used. To do this we will first continue to assume that all sequence numbers are integers increasing without bound. We must then show that all SD sequence number parameters at the transmitter, including those contained in arriving STATs and USTATs, fall between  $VT(A)$  and  $VT(A) + 2^{24} - 1$ , inclusive. All SD sequence number parameters at the receiver, including those contained in arriving SDs and POLLs, must fall between  $VR(R)$  and  $VR(R) + 2^{24} - 1$ , inclusive, and all POLL sequence numbers at the transmitter must fall between  $VT(PA)$  and  $VT(PA) + 2^{24} - 1$ , inclusive. To indicate the value of a parameter at time  $t$ , we will append the suffix “. $t$ ”.

The most interesting point that is proved below is that with a modulus of  $2^{24}$ , the receive window can be as large as  $2^{24} - 1$  without causing ambiguity with sequence numbers. The receive window is allowed to be this large due to the fact that there are no unnecessary retransmissions. In general selective repeat systems, where unnecessary retransmissions can occur, if the modulus is  $M$ , then the receive window is limited to size  $M/2$  [1].

### 3.5.1. Parameters at the receiver

Assume sequence numbers are integers increasing without bound, and consider the successful transmission of an arbitrary SD sent by the source. Assume it is transmitted at time  $t_1$  and received at the destination at time  $t_2$ . ( $t_2$  is the instant of arrival, before any updating of variables occurs.) Obviously,  $t_2 \geq t_1$ . The sequence number of the SD, say  $X$ , must lie in the transmitter’s send window at time  $t_1$ . Thus

$$VT(A).t_1 \leq X \leq VT(MS).t_1 - 1. \quad (1)$$

At any time  $t$ ,  $VR(MR).t$  can never be larger than  $VR(R).t + 2^{24} - 1$  (i.e., the receive window can be no larger than  $2^{24} - 1$ ). Thus, since  $VT(MS)$  is updated via the value of  $VR(MR)$  carried in STATs and USTATs, and since  $VR(R)$  is non-decreasing in  $t$ :

$$\begin{aligned} VT(MS).t_1 &\leq VR(R).t_1 + 2^{24} - 1 \\ &\leq VR(R).t_2 + 2^{24} - 1. \end{aligned} \quad (2)$$

In the previous section it was shown that the protocol does not produce unnecessary retransmissions. Thus, the transmission of SD  $X$  must be necessary, i.e.,  $X \geq V(R).t_2$ . (If  $X$  were less than  $V(R).t_2$ , it would mean SD  $X$  had already been received by time  $t_2$ .) Combining this with equations (1) and (2), yields

$$VR(R).t_2 \leq X \leq VR(R).t_2 + 2^{24} - 2. \quad (3)$$

Thus,  $X$  falls within the desired range.

Next, consider a POLL that is transmitted at time  $t_1$  and received at the destination at time  $t_2$ .  $POLL.N(S) - 1$  is the highest numbered SD sent before the POLL. Assume the SD with sequence number  $POLL.N(S) - 1$  was transmitted at time  $t_0$ , where  $t_0 \leq t_1 \leq t_2$ . Due to the restriction on the send window, and using (2):

$$\begin{aligned} POLL.N(S) - 1 &< VT(MS).t_0 \\ &\leq VR(R).t_0 + 2^{24} - 1 \\ &\leq VR(R).t_2 + 2^{24} - 1. \end{aligned} \quad (4)$$

Since PDUs travel in order, the SD with sequence number  $POLL.N(S)$  cannot possibly arrive before a POLL containing  $POLL.N(S)$ ; thus, at time  $t_2$ ,

$$VR(R).t_2 \leq POLL.N(S). \quad (5)$$

Combining Eqs. (4) and (5) yields

$$VR(R).t_2 \leq POLL.N(S) \leq VR(R).t_2 + 2^{24} - 1. \quad (6)$$

Finally, we need to check the value of  $VR(H)$  at the arrival time  $t_2$  of an SD or POLL. From the definition of the protocol,  $VR(H) \geq VR(R)$  for all  $t$ . Also, from the definition of the protocol,  $VR(H) \leq VR(MR)$  for all  $t$ . Due to the restriction on the size of the receive window, we know that

$$VR(MR).t_2 \leq VR(R).t_2 + 2^{24} - 1. \quad (7)$$

Thus,

$$VR(R).t_2 \leq VR(H).t_2 \leq VR(R).t_2 + 2^{24} - 1. \quad (8)$$

Overall, we have shown that at the arrival time of an SD or POLL, the value of  $VR(H)$ ,  $VR(MR)$ , the SD sequence number, and  $POLL.N(S)$  all lie between  $VR(R)$  and  $VR(R) + 2^{24} - 1$ , inclusive.

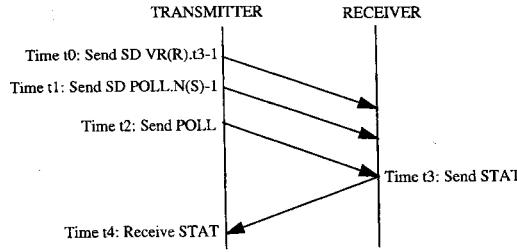


Fig. 11. A POLL is sent at time  $t_2$ , which triggers a STAT at time  $t_3$ . The value of  $VR(R)$  at time  $t_3$  is  $VR(R).t_3$ , which indicates that the SD with sequence number  $VR(R).t_3 - 1$  must have arrived at the receiver prior to time  $t_3$ .

### 3.5.2. Parameters at the transmitter

Again, assume increasing integers are used for all sequence numbers. It will be shown that  $VT(S)$ ,  $VT(MS)$  and any sequence numbers contained in STATs or USTATs fall within a proper range at the transmitter so that sequence numbers can be treated modulo  $2^{24}$ , where the base is  $VT(A)$ . We will begin by showing that at the time of a STAT's arrival at the transmitter,  $STAT.N(R)$  falls between  $VT(A)$  and  $VT(A) + 2^{24} - 1$ , inclusive. (The argument for USTATs is almost identical to that for STATs; in the steps below, replace  $POLL.N(S)$  by the sequence number of the SD generating the USTAT.)

Consider any STAT that arrives at the source at time  $t_4$ , where  $t_4$  is prior to any processing of the STAT being done. Assume the STAT was sent by the receiver at time  $t_3$ , and assume that the POLL that triggered this STAT was sent by the transmitter at time  $t_2$ , where  $t_2 \leq t_3 \leq t_4$  (refer to Fig. 11).  $STAT.N(R)$  equals the value of  $VR(R)$  at time  $t_3$ . Since  $VR(R)$  is non-decreasing in  $t$  and since PDUs travel in sequence, it must be true that

$$VT(A).t_4 \leq STAT.N(R). \quad (9)$$

(If this were not true, then the transmitter would have previously received an ACK for an SD that was not ACKed in this STAT.)

From the initialization statement of the protocol, it holds that at the start of the algorithm we have

$$VT(MS) \leq VT(A) + 2^{24} - 1. \quad (10)$$

$VT(MS)$  is only updated when a STAT or USTAT arrives. We will assume that (10) holds for all time

up to time  $t_4$ , and then show below that it must always hold.

The SD with sequence number  $VR(R).t_3 - 1$  must have been received at the destination by time  $t_3$ , and thus, must have been sent by the transmitter at some time  $t_0$ , where  $t_0 \leq t_3$ . Combining the restriction on the send window with (10) and the nondecreasing property of  $VT(A)$  yields

$$\begin{aligned} VR(R).t_3 - 1 &< VT(MS).t_0 \\ &\leq VT(A).t_0 + 2^{24} - 1 \\ &\leq VT(A).t_4 + 2^{24} - 1. \end{aligned} \quad (11)$$

Combining (9) and (11), and using the fact that  $STAT.N(R)$  equals  $VR(R).t_3$ , we see that

$$VT(A).t_4 \leq STAT.N(R) \leq VT(A).t_4 + 2^{24} - 1. \quad (12)$$

Thus,  $STAT.N(R)$  falls within the desired range.

After the STAT arrives,  $VT(A)$  is updated to the value  $STAT.N(R)$ . We will refer to the updated value of  $VT(A)$  as  $VT(A).t_4^+$ . When considering the sequence numbers modulo  $2^{24}$ ,  $VT(A).t_4^+$  will be the new "base" at the transmitter.

Since the maximum sized receive window is  $2^{24} - 1$ , we know that

$$VR(R).t_3 \leq VR(MR).t_3 \leq VR(R).t_3 + 2^{24} - 1. \quad (13)$$

and thus, after  $VT(A)$  is updated to  $STAT.N(R)$  (which equals  $VR(R).t_3$ ), we have

$$\begin{aligned} VT(A).t_4^+ &\leq VR(MR).t_3 \\ &\leq VT(A).t_4^+ + 2^{24} - 1. \end{aligned} \quad (14)$$

Since  $STAT.N(MR)$  equals  $VR(MR).t_3$ , the bounds in (14) hold for  $STAT.N(MR)$  also.  $VT(MS)$  is updated to  $STAT.N(MR)$  when a STAT arrives, thus:

$$VT(A).t_4^+ \leq VT(MS).t_4^+ \leq VT(A).t_4^+ + 2^{24} - 1. \quad (15)$$

Since  $VT(MS)$  is only updated when STATs or USTATs arrive, this shows that (10) will hold for all time  $t$ .

Next, we need to consider any NACKs in the STAT. Since the POLL containing  $POLL.N(S)$  was

sent at time  $t_2$ , the SD with sequence number  $POLL.N(S) - 1$  must have been sent at some time  $t_1$ , where  $t_1 \leq t_2$ . From the restriction on the send window, we know that

$$POLL.N(S) \leq VT(MS).t_1. \quad (16)$$

From the algorithm for sending a STAT, we know that any SD NACKed in the STAT satisfies

$$VR(R).t_3 \leq NACK < POLL.N(S). \quad (17)$$

From (16) and (17), and the fact that  $VT(A).t_4^+$  equals  $VR(R).t_3$ :

$$VT(A).t_4^+ \leq NACK < POLL.N(S) \leq VT(MS).t_1. \quad (18)$$

Using (10) and the nondecreasing property of  $VT(A)$  implies

$$\begin{aligned} VT(A).t_4^+ &\leq NACK < VT(MS).t_1 \\ &\leq VT(A).t_1 + 2^{24} - 1 \\ &\leq VT(A).t_4^+ + 2^{24} - 1. \end{aligned} \quad (19)$$

Lastly, we need to consider  $VT(S)$  at any time  $t$ , where  $VT(S)$  is the sequence number of the next SD to be sent for the first time. From the definition of the protocol, it must hold that  $VT(A).t \leq VT(S).t$  for all time  $t$ . Assume the SD with sequence number  $VT(S).t - 1$  was sent at time  $t'$ , where  $t' \leq t$ . Due to the restriction on the send window, then

$$VT(S).t \leq VT(MS).t'.$$

Using (10),

$$\begin{aligned} VT(A).t &\leq VT(S).t \leq VT(MS).t' \\ &\leq VT(A).t' + 2^{24} - 1 \\ &\leq VT(A).t + 2^{24} - 1. \end{aligned}$$

Thus, for all time  $t$ ,  $VT(S)$  falls between  $VT(A)$  and  $VT(A) + 2^{24} - 1$ , inclusive.

Overall, we have shown that any SD sequence numbers maintained at the transmitter or contained in a STAT (or USTAT) fall within the desired range with respect to  $VT(A)$ . Combining this section and the previous section, we have shown that SD sequence numbers can be kept modulo  $2^{24}$  without affecting the operation of the protocol at the transmitter or the receiver.

### 3.5.3. Poll sequence numbers

In this section, we show that POLL sequence numbers can be maintained modulo  $2^{24}$  without affecting the POLL sequence number comparisons that must be performed when STATs arrive.  $VT(PA)$  is used as the “base” when performing comparisons modulo  $2^{24}$ , where  $VT(PA)$  is the sequence number of the last received STAT.

First consider POLL sequence numbers as ordinary increasing integers. As specified in the algorithm for transmitting POLLS, the “send window” for POLLS is of size  $2^{24} - 1$ , so the next STAT to arrive must always contain a  $STAT.N(PS)$  that falls between  $VT(PA) + 1$  and  $VT(PA) + 2^{24} - 1$ , inclusive (this is prior to  $VT(PA)$  being updated due to the STAT arrival). At the arrival time of the STAT, due to the non-decreasing property of  $VT(PS)$ , the current value of  $VT(PS)$  must lie between  $STAT.N(PS)$  and  $VT(PA) + 2^{24} - 1$ , inclusive.

After a STAT arrives, the SDs in the transmission buffer fall into one of the following categories:

- SD was ACKed, so no further comparisons with POLL sequence numbers are necessary.
- SD was NACKed but not retransmitted since its POLL stamp was greater than or equal to  $STAT.N(PS)$ . Due to the POLL send window, its POLL stamp must be less than or equal to  $VT(PA) + 2^{24} - 1$ .
- SD was NACKed and retransmitted. After the SD is retransmitted, its POLL stamp will be updated to the current value of  $VT(PS)$ , which falls between  $STAT.N(PS)$  and  $VT(PA) + 2^{24} - 1$ , inclusive.
- SD was not NACKed or ACKed. Thus, it must have been transmitted after the POLL that triggered the STAT, so that its POLL stamp must be greater than or equal to  $STAT.N(PS)$ . Due to the POLL send window, its POLL stamp must be less than or equal to  $VT(PA) + 2^{24} - 1$ .

In the initialization step of the protocol,  $VT(PA)$  is set to 0; thus, initially, all POLL sequence number stamps are greater than or equal to  $VT(PA)$ . After a STAT arrives and is completely processed,  $VT(PA)$  is updated to  $STAT.N(PS)$ , and as shown above, all POLL stamps in the transmission buffer will be greater than or equal to  $STAT.N(PS)$ . (There may be ACKed SDs left in the buffer with POLL stamps less than  $STAT.N(PS)$ , but no further POLL stamp com-

parisons are necessary for ACKed SDs.) Thus, all relevant poll stamps will continue to fall between  $VT(PA)^+$  and  $VT(PA)^+ + 2^{24} - 1$ , inclusive, where  $VT(PA)^+$  is the updated value of  $VT(PA)$ . This implies that POLL sequence numbers can be maintained modulo  $2^{24}$  without ambiguity, where  $VT(PA)^+$  serves as the base.

In the protocol definition in [5],  $VT(PA)$  is updated to  $STAT.N(PS)$  prior to checking the POLL stamps of the NACKed SDs. With  $VT(PA)^+$  as the base for the modulo arithmetic, all POLL stamps would be considered greater than or equal to  $VT(PA)^+$ , resulting in no retransmissions. In the algorithm provided in Section 2.2, this is corrected, and  $VT(PA)$  is updated after the POLL stamp comparisons are performed.

#### 4. Conclusions

We have proved that under certain conditions, the ATM Adaptation Layer retransmission scheme will generate retransmissions of lost frames without producing unnecessary retransmissions (assuming some corrections to the specifications in [5] are made). It satisfies the conditions of both safety and liveness. Due to the property of no unnecessary retransmissions, the receive window can be as large as  $2^{24} - 1$  if the sequence numbers are maintained modulo  $2^{24}$ .

#### References

- [1] D. Bertsekas and R. Gallager, *Data Networks* (Prentice-Hall, Englewood Cliffs, NJ, 2nd ed., 1992).
- [2] G. Bochmann and C. Sunshine, A survey of formal methods, in: P. Green, ed., *Computer Network Architectures and Protocols* (Plenum Press, New York, 1982).
- [3] B. Halpern and S. Owicki, Modular verification of computer communication protocols, *IEEE Trans. Commun.* **31** (1) (1983).
- [4] ITU-T recommendation I.363, B-ISDN ATM Adaptation Layer (AAL) specification, Geneva, 1992.
- [5] ITU-T Study Group 11, Draft new ITU-T recommendation Q.2110: B-ISDN ATM Adaptation Layer – Service Specific Connection Oriented Protocol (SSCOP), Report R30, January 1994.
- [6] ITU-T Study Group 11, Draft new ITU-T recommendation Q.2130: B-ISDN signalling ATM Adaptation Layer – Service specific coordination function for support of signalling at the user-to-network interface, Report R31, January 1994.
- [7] ITU-T Study Group 11, Draft new ITU-T recommendation Q.2140: B-ISDN signalling ATM Adaptation Layer – Service specific coordination function for support of signalling at the network node interface, Report R70, September 1994.
- [8] M. Schwartz, *Telecommunication Networks* (Addison-Wesley, Reading, MA, 1987).