# Adaptive Acknowledgment Scheme for Efficient Error Control in ATM Clustering System

Jong-Kwon Lee, Yong Jae Kim, Jae Woong Chung, and Tag Gon Kim
Department of Electrical Engineering,
Korea Advanced Institute of Science and Technology (KAIST),
373-1 Kusong-dong, Yusong-gu, Taejon 305-701, Korea
{jklee,yjkim,jwchung,tkim}@coregate.kaist.ac.kr

## Abstract

*An ATM clustering system is a kind of workstation clusters over an ATM network. Such a system can be used as a distributed database server which requires reliable data delivery. This paper proposes an error recovery scheme at the transport layer for reliable data transfers with high throughput in the ATM clustering system. For such data transfers, acknowledgments are sent periodically as well as at the detection of packet losses. To be efficient, the periods of acknowledgments for the scheme are adaptively varied, depending on whether current transfer is either in error free phase or in an error recovery phase. The design and performance evaluation of the error recovery scheme is presented with a set of parameter values to achieve the best performance. The results show that the proposed scheme operates correctly and efficiently for the ATM clustering system. The throughput performance of the proposed scheme is superior to the SSCOP (Service Specific Connection Oriented Protocol) regardless of packet loss rates.*

## 1. Introduction

Some recent works have been devoted to build and evaluate workstation clusters over ATM networks. [5] has shown that among four different API's a distributed program using the ATM API can obtain a better communication performance than the others. [8] have regarded a workstation cluster over an ATM network a platform of High Performance Distributed Computing(HPDC). An ATM clustering system is a kind of workstation clusters used as a distributed database server. The system consists of several nodes each of which is configured with a 155 Mbps adapter board. User data can be stored in each node by using either partitioned or replicated schemes [7]. In such a system, *reliable delivery* should be supported to achieve data integrity whatever the scheme is. When the system processes a query, especially join operation, local data at each node may or should be exchanged to get a correct result. The size of the data may be large, and the traffic may be very bursty, which may result in an overflow at the ATM switch buffers. Because there is no end-to-end reliable delivery service at the ATM layer, data loss due to such an overflow results in an incorrect, useless output.

Several protocols have been suggested and developed for reliable data transfers in the generic ATM networks. SNR is designed as a general purpose transport protocol [2, 6]. A distinctive feature of SNR is the periodic exchange of state information between the transmitter and the receiver. Retransmission requests are selective on a block basis, so that the requests are not sent immediately after the detection of packet losses, but only periodically. This feature may degrade throughput as the period is long, because of the transmitter's slow reaction to packet losses. SSCOP (Service Specific Connection Oriented Protocol) [3] has been influenced by the SNR design. Although the SSCOP has been recommended by the ITU-T for supporting signaling, it may be used to provide a generic reliable data transfer service. However, SSCOP has many PDU types which are used for data transfers as well as for signaling. SSCOP uses a polling mechanism to exchange state information periodically. Selective retransmission is also requested at the detection of packet losses. For the specific use of SSCOP in the ATM-based workstation clusters, [8] has proposed two cell-based mechanisms. It is suited for HPDC applications, but not for database applications.

To support reliable data transfers in the ATM clustering system, we propose and evaluate an error recovery protocol specific to the system. We can simplify the protocol for the following reasons: the network is connection oriented, and bandwidth-delay product is very small for short communication links between nodes. To transfer burst data, we use an UBR service category, which does not guarantee any

QoS parameters and is best suited for general data transfer. In the proposed scheme, acknowledgments are reported not only at the detection of errors but also periodically. A smart feature of the scheme is that the frequencies of the acknowledgments are varied according to the conditions of communication channels which may be either under normal or under error phases. We also present the performance evaluation of our scheme compared with the SSCOP.

This paper is organized as follows. In Section 2, we describe the proposed error recovery protocol for the ATM clustering system. In Section 3, performance evaluation of the proposed error recovery protocol are presented. Finally, we conclude this paper in Section 4.

# 2. Design of Error Recovery Protocol

## 2.1. Design Principles of the Proposed Error Recovery Protocol

Our target system, the ATM clustering system, uses an ATM local-area network. It is based on a connection-oriented network in which packets are assumed to be delivered in sequence. This leads to the fact that gaps in the packet sequence numbers mean losses of the corresponding packets due to overflows in buffers or bit errors in fiber lines. With this fact we can simplify error control protocol and use fast error recovery mechanisms for the ATM clustering system.

As described in [1], the design of light-weight protocols puts much effort to minimize the number of packet exchange, and especially to shorten the length of the instruction path for data transmission and reception when no errors occur. Such effort may also be required to the design of error recovery protocol for the ATM clustering system.

In addition, we should determine the packet size which is the unit of (re)transmission and error control. From the viewpoint of packet processing in the terminal nodes, there are two kinds of overhead, overhead per each packet and memory access (i.e., overhead per byte). Small packet size has relatively more overhead per packet for the same size of data to be transferred. Thus, large packet size is preferred. However, if the packet size is large, the overhead for retransmission may increase. From the above two aspects, it remains as a design trade-off to determine the packet size, which may be optimized by simulation and experiments after implementation.

For error detection, there is no checksum in the proposed error recovery protocol of transport layer because underlying AAL5 is already responsible for the checksum. Error detection is only by detecting gaps between sequence numbers.

We could develop our protocol for generic data transfers so that the overhead of protocol processing is minimized by reducing the number of packet types and using more efficient error control algorithm. In developing our protocol, we have been concerned with error recovery procedures only in a data transfer phase by ignoring a connection-establishment/release phase.

## 2.2. Packet Formats and Protocol Operations

We simplified packet formats into only two kinds, DATA packet and ACK packet, being concerned with only data transfers. Figure 1 shows the packet formats.

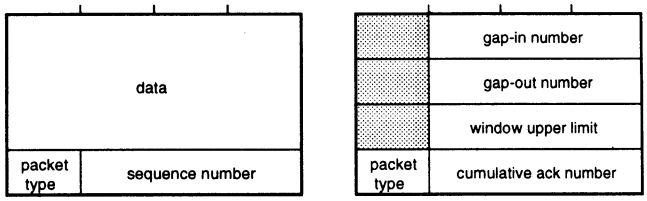


**Figure 1. Packet Formats.**

The DATA packet is responsible for delivering data as a payload from the upper layer at a source node to a destination node. *Sequence number* is the number with which each packet can be identified during data transfers.

The ACK packet consists of five fields including a packet type field. *Window upper limit* is the last sequence number of packets that the transmission buffer can accommodate at that time. The transmitter can send packets up to this number without receiving any acknowledgments from the receiver. *Cumulative ack number* represents the sequence number of the next in-sequence packet expected to be received at the receiver. For example, suppose the receiver has received packets from 1 to 9, 13, and 16. Then the *cumulative ack number* becomes 10. *Gap-in number* and *gap-out number* respectively represent the first number and the next to the last number of a gap in the sequence space when new packet losses are detected, or when the status information of the receiver is periodically reported. For the above example, when the packet 13 is received, the receiver makes a ACK packet with a pair (*gap-in number*, *gap-out number*) = (10,13), and when the 16th packet is received, it makes a pair (14,16) within a ACK packet. In each case, the receiver sends the ACK packet to the transmitter immediately. If the timer requires to send the status information, the receiver checks gaps unfilled by that time and sends a set of ACK packets one after another.

Timers are instrumented in error control protocols. Some protocols have timers at the transmitter side, and others at the receiver side. In case of SSCOP, timers are at the transmitter, and one of them causes the transmitter to request the

status information of the receiver. In our protocol, timers are at the receiver side and cause the receiver to spontaneously send its status information to the transmitter. This is one of the differences between our protocol and SSCOP.

Suppose that a connection between the transmitter and the receiver has already been set up. Then, we will explain the error control operations separately at the transmitter and the receiver.

*1) Transmitter Operations:* The transmitter has three main tasks related to error control. First, as a fundamental task, it transmits new packets whenever there is data in the transmission queue. The transmission queue is serviced in the order of increasing sequence numbers whenever the bandwidth is available. Every data unit in the transmission queue is put into a new DATA packet as a payload. Then, the current value of a sequence number is copied into the *sequence number* field, and the counter for sequence numbers is increased by one. This newly constructed packet is passed down to the lower layer (such as AAL5) and saved in the transmission buffer until acknowledged by the receiver.

Secondly, the transmitter processes retransmission requests and schedules requested packets for transmission. The retransmission requests are included in the ACK packets sent by the receiver. *Gap-in number* and *gap-out number* are used to determine which packets should be retransmitted. If the two are equal, no packets in the transmission buffer are retransmitted. When they are not equal, the corresponding packets from *gap-in number* up to *gap-out number* $-$ 1 are put into the retransmission queue. The packets in the retransmission queue are sent prior to data units in the transmission queue. When the retransmission queue is empty, the transmitter resumes sending the next data unit in the transmission queue.

The third task of the transmitter is to release the packets in the transmission buffer which are acknowledged by *cumulative ack number* in the received ACK packets. All the packets with sequence numbers below this number are released from the buffer. This operation may be considered as a simple sliding window mechanism.

*2) Receiver Operations:* The main task of the receiver is to perform per-packet processing which includes detecting gaps in the sequence numbers. Since a connection-oriented network is assumed, the gaps mean losses of the corresponding packets. When the receiver detects packet losses, it requests retransmission by setting *gap-in number* and *gap-out number* of an ACK packet.

Another important task of the receiver is to report its status information to the transmitter whenever the periodic timer is expired. The status information is in the form of an ACK packet within which current status about packet arrivals is designated. When there are no undelivered packets, *gap-in number* and *gap-out number* are set equal to *cumulative ack number*. However, if some packets are still in loss, the receiver requests retransmission by sending a set of ACK packets each of which corresponds to a gap.

The interval of the timer with which the receiver reports its status information can be varied between two values, $T_{normal}$ and $T_{error}$. $T_{normal}$ is for the normal situation, i.e., the error-free phase, and $T_{error}$ for the error phase in which there are lost packets. Therefore, the period of status information report can be adaptive in accordance with the current communication conditions. Such an adaptive scheme is an advantage of our protocol compared to SSCOP in which the transmitter polls the receiver with a fixed period dependent on applications. The two timer intervals of our protocol may be defined either based on connection by connection or based on applications.

# 3. Evaluation of the Proposed Error Recovery Scheme

## 3.1. Correctness of the Protocol Operations

To show the correctness of our protocol operations, we will present error recovery procedures for two cases: a DATA packet loss case and an ACK packet loss case.

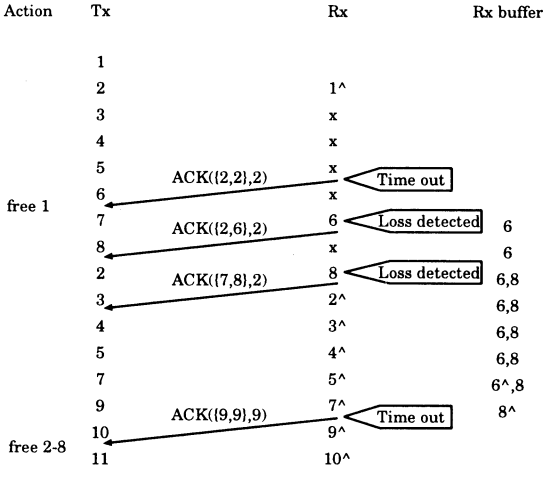


**Figure 2. Protocol operation with a bursty loss of DATA packets.**

Figure 2 illustrates the first case in which some DATA packets transmitted to the receiver experience bursty losses. In the case, a list of numbers beside each ACK packet represents ({*gap-in number*, *gap-out number*}, *cumulative ack number*), and the symbol '^' means the packets marked with this are delivered to the upper layer.

The first ACK packet is sent with lists ({2,2},2), because the timer is expired and the only one packet with sequence number 1 is received. If the transmitter receives this ACK

packet, it releases the DATA packet 1 from the transmitter buffer.

Suppose DATA packets between sequence numbers 2 and 5 have been lost somewhere in the network. The receiver then receives DATA packet 6 and guesses that the packets between 2 and 5 must be lost. It immediately sends the transmitter an ACK packet which requests retransmission of the lost packets. Consecutively, if DATA packet 8 has been received, DATA packet 7 is regarded as being lost and a retransmission of this packet is requested.

The timer value has been changed to $T_{error}$ after the first error occurred, and this value will not be returned to $T_{normal}$ until all the lost packets are recovered. The fourth ACK packet in this case was sent when the timer was expired with an interval $T_{error}$. Then, the timer is again set to $T_{normal}$ since all the errors have been recovered.

Figure 3 shows the second case where an ACK packet is lost following a DATA packet loss. The ACK packet loss can be overcome by periodic status informations reported from the receiver to the transmitter. Our protocol has also been verified to be correct for other error scenarios by using SDT, a commercial tool supporting SDL description [9].

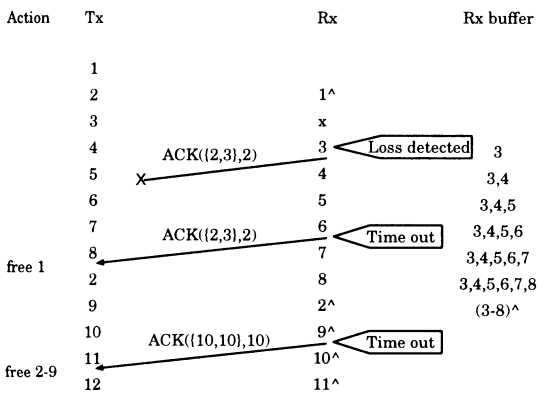


**Figure 3. Protocol operation with a ACK packet loss.**

## 3.2. Performance Analysis

Discrete event simulation was applied to evaluate the performance of our protocol. Simulation was carried out by using the DEVSim++ simulation environment [4], which is a C++ realization of the DEVS (Discrete Event Systems Specification) formalism [10].

For our simulation objectives, the overall system model is composed of a underlying network and end systems (see Figure 4). The underlying network is modeled as a black box characterized by a channel bandwidth and a packet loss process. A propagation delay can be ignored because the ATM clustering system is a local-area network. Two transmission models in the underlying network are data and acknowledgment channels. They incorporate with the lower layer queues of each end system. The end system consists of two transport entities, a transmitter and a receiver. Generator generates input data to be transmitted by the transmitter, and Transducer collects the delivered data in the order of the sequence number. Transducer also analyzes statistical information for performance measure. This protocol model is simple but accurate enough to evaluate the performance of transport protocols.

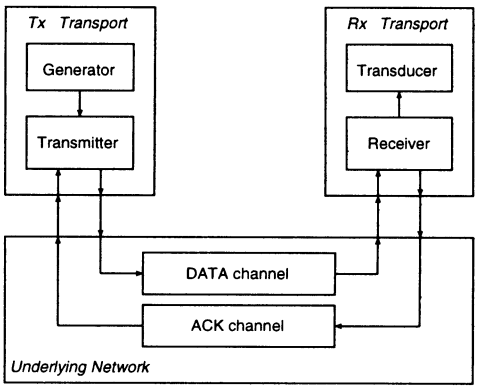


**Figure 4. Simulation model of transport protocols.**

| Parameter | | | Value |
|---|---|---|---|
| End systems | Packet size | DATA | 1024 Bytes |
| | | ACK | 32 Bytes |
| | Window size of Tx buffer | | varied |
| | $T_{normal}$ | | varied |
| | $T_{error}$ | | varied |
| | Processing Overhead | per byte | 0.05 $\mu s$ |
| | | per packet | 70 $\mu s$ |
| Underlying network | Bandwidth | DATA channel | 100 Mbps |
| | | ACK channel | 10 Mbps |
| | Packet loss type | | Uniform or Bursty |
| | Packet loss rate | | varied |
| | Propagation delay | | $\approx 0$ |

**Table 1. Simulation parameters.**

### 3.2.1. Simulation Setup

Table 1 summarizes simulation parameters used for performance evaluation of our protocol. As shown in the table, some parameters are fixed while others are varied.

We use a performance index, *throughput efficiency*, which is defined as the ratio of the actual throughput under error condition and the ideal throughput under error-free condition.

A set of simulation runs has been carried out for each combination of parameters, which makes the experiments more reliable. Simulation run-length for each experiment was adjusted to reflect a packet loss rate. For example, if a packet loss rate is 0.1, Transducer collects ten thousand of packets delivered from the Receiver model, which experiences losses of a thousand packets.

### 3.2.2. Simulation Results

Figures 5 and 6 show the simulation results for various combinations of two timer intervals, $T_{normal}$ and $T_{error}$, with the low packet loss rates of $p = 10^{-4}$ and $p = 10^{-3}$, respectively. Most packet losses for these loss rates may arise from bit errors of transmission lines. Since the ATM clustering system may operate mostly under normal situations with the packet loss rate between $10^{-4}$ and $10^{-3}$, it is necessary to determine a set of parameters for which the system operates well under low-error conditions. In this low-error case, the effect of $T_{error}$ on the throughput is little as shown in the figures. Note that the throughput increases as $T_{normal}$ increases. However, it becomes smaller after a certain value of $T_{normal}$. With a small $T_{normal}$, the window remains available; throughput becomes higher as an acknowledgment interval becomes longer. But when $T_{normal}$ becomes large enough to fill the window with unacknowledged packets, throughput is decreased. As shown in Figures 5 and 6, simulation experiments suggest an optimal value of $T_{normal}$, 40 ms for the window size of 400.

Figure 7 shows throughput variations for various combinations of two timer intervals when the packet loss rate is high ($p = 10^{-1}$). In this case, $T_{error}$ rather than $T_{normal}$ mainly affects the throughput. The reason is that if DATA packets are lost very frequently, most ACK packets are sent with the interval $T_{error}$.

The above two experiments with different packet loss rates suggest optimal values of two parameters, $T_{normal}$ and $T_{error}$, to be 40 ms and 5 ms, respectively. With the two values the throughput against window size have been measured. Figure 8 shows the result. From this result, we have concluded that for relatively large window sizes the throughput decreases as the packet loss rate becomes higher, on the other hand, for small window sizes the throughput can increase when the packet loss rate is reasonably high. This is because our protocol uses $T_{error}$ as a timer interval in error phases. Therefore, for a loss rate of $10^{-2}$, a small value of $T_{error}$ releases acknowledged packets from the window of the transmission buffer more frequently, resulting in a higher throughput than for the loss rates under $10^{-2}$. From these experiments, a maximum throughput efficiency with a window size of 400 requires the timer intervals of $T_{normal}$ and $T_{error}$ to be 40 ms and 5 ms, respectively.

To show the excellence of our protocol, we have compared the throughput efficiency of ours with that of SSCOP. For that, we need to obtain an optimal value of $Timer\_POLL$ of SSCOP with which the transmitter polls the receiver's status. This optimal timer interval was different for the loss rates, i.e., it was 40 ms for low error rates and 10 ms for high error rates. Therefore, the throughput efficiency of SSCOP for both cases was measured and compared with that of our protocol for various packet loss rates. In addition, we have also varied loss types between uniform and bursty losses (Figures 9 and 10). Both comparisons show that our protocol is superior to SSCOP in the throughput efficiency regardless of packet loss rates.

## 4. Conclusions

This paper considers an error recovery problem in an ATM clustering system used as a distributed database server. The ATM clustering system requires reliable delivery of user data between source and destination nodes. With the investigation of features of the ATM clustering system, we could notice that a more efficient and light-weight error recovery scheme would be necessary for such a system.

This motivation has led us to develop a simple adaptive error control scheme for the ATM clustering system. This scheme has inherited, from the SSCOP, a feature that the transmitter is informed of the receiver's status periodically as well as at the detection of packet losses. One of the differences between the two schemes, however, is that our scheme sends the receiver's status without polling from the transmitter. Besides, it uses two timer intervals to report the status with less overheads, while the SSCOP uses only one timer value to poll the receiver's status. The use of two timer intervals makes our scheme adaptive simply to the communication circumstances, i.e., under normal or error conditions of communication channels.

We have presented the simulation results about the performance of our protocol in throughput efficiency. From the results, we could obtain a combination of parameters to achieve the best performance. The results has also shown that our retransmission scheme gives ã superior throughput performance to those of the SSCOP retransmission scheme for low- and high-error conditions.

To determine the optimal set of parameter values for the real system, it is necessary to implement the proposed error recovery scheme and evaluate the performance based on the implementation. This may be done in future works.

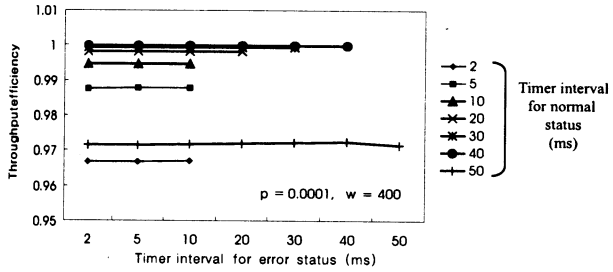


**Figure 5. Window size = 400, packet loss rate = $10^{-4}$ (BER = $10^{-8}$), uniform loss type.**

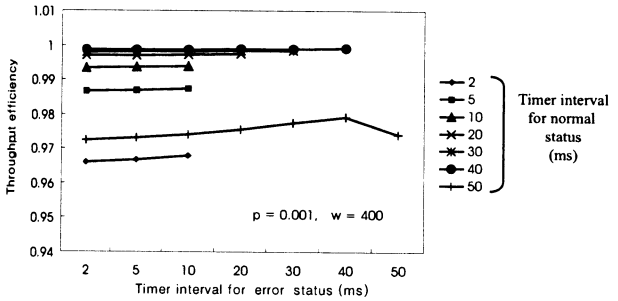


**Figure 6. Window size = 400, packet loss rate = $10^{-3}$ (BER = $10^{-7}$), uniform loss type.**

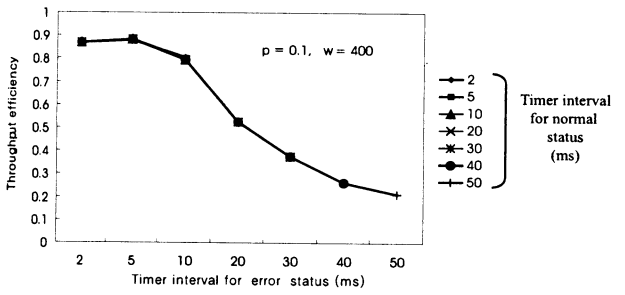


**Figure 7. Window size = 400, packet loss rate = $10^{-1}$, uniform loss type.**

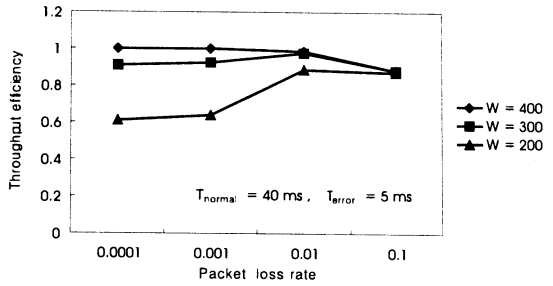


**Figure 8. Window size variation; $T_{normal}$ = 40 ms, $T_{error}$ = 5 ms, uniform loss type.**

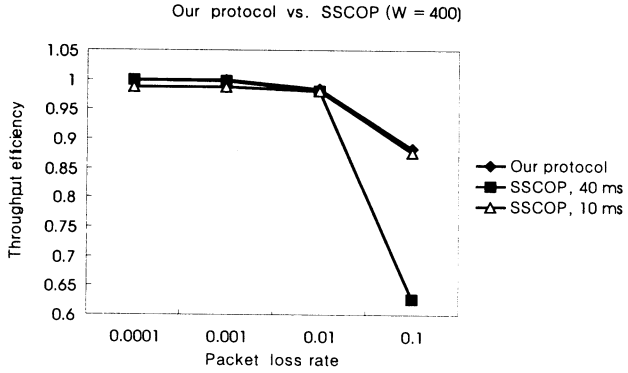


**Figure 9. Comparison with SSCOP; $T_{normal}$ = 40 ms, $T_{error}$ = 5 ms, $Timer\_POLL$ = 10 ms and 40 ms, uniform loss type.**

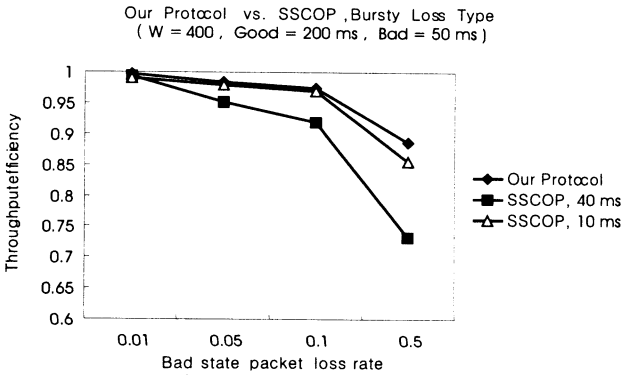


**Figure 10. Comparison with SSCOP; $T_{normal}$ = 40 ms, $T_{error}$ = 5 ms, $Timer\_POLL$ = 10 ms and 40 ms, bursty loss type.**

## References


[1] W. A. Doeringer *et al.* A survey of light-weight transport protocols for high-speed networks. *IEEE Trans. Commun.*, 38(11):2025–2039, Nov. 1990.

[2] B. T. Doshi *et al.* Error and flow control performance of a high speed protocol. *IEEE Trans. Commun.*, 41(5):707–720, May 1993.

[3] ITU-T Recommendation Q.2110. *B-ISDN ATM Adaptation Layer – Service Specific Connection Oriented Protocol (SSCOP).* ITU, Geneva, July 1994.

[4] T. G. Kim. *DEVSim++ User's Manual: C++ Based Simulation with Hierarchical, Modular DEVS Models.* Systems Modeling Simulation Lab., KAIST, Taejon, Korea, 1994. ftp://sim.kaist.ac.kr/pub/DEVSim++-1.0/.

[5] M. Lin *et al.* Distributed network computing over local ATM networks. *IEEE J. on Select. Areas Commun.*, 13(4):733–748, May 1995.

[6] A. N. Netravali, W. D. Roome, and K. Sabnani. Design and implementation of a high-speed transport protocol. *IEEE Trans. Commun.*, 38(11):2010–2024, Nov. 1990.

[7] M. T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems.* Prentice-Hall International, Inc., 1991.

[8] J. Sole-Pareta and J. Vila-Sallent. Network-based parallel computing over ATM using improved SSCOP protocol. *Computer Communications*, 19:915–926, 1996.

[9] Telelogic. *SDT Technical Presentation.* Telelogic, 1997.

[10] B. P. Zeigler. *Multifacetted Modeling and Discrete Event Simulation.* Academic Press, Orlando, FL, 1984.