Qatar Univ. Sci. J. (1994), 14 (1): 23-30

DESIGN AND PERFORMANCE EVALUATION OF A HIGH SPEED TRANSPORT PROTOCOL

By

M. N. EL-DERINI* and R. M. GUIRGIS**
* Department of Computer Science, University of Qatar, Doha, Qatar
** Computer Science & Automatic Control Dept., Faculty of Engineering, Alexandria University

تصميم وتقييم أداء بروتوكول انتقال عالي السرعة

محمد نزيه الدريني و رامي مكرم جرجس

لا تتناسب بروتوكولات الانتقال المصممة للشبكات الحالية لشبكات المستقبل ذات السرعات العالية وذلك لزيادة سعة تردد الشبكة بالنسبة لسرعات مكونات المعالج . وتعتبر الشبكات ذات السرعات العالية التي تصل من ١٠٠ مليون وحدة ثنائية في الثانية إلى واحد بليون وحدة ثنائية في الثانية متاحة الآن ولكن السرعة لا تتعدى ١٠ مليون وحدة ثنائية في الثانية أعلى طبقة الانتقال بالنسبة للبروتوكولات الحالية . وذلك زاد من الاحتياج وتطبيق بناءات وبروتوكولات جديدة .

وقد قمنا في هذا البحث بوضع نموذج لبروتوكول الانتقال وتطبيق فكرة تبادل المعلومات الكاملة عن الحالة بين المرسل والمستقبل على فترات متكررة . وتم تطبيق نسختين للمحاكي إحداهما تطبق طريقة الاستجابة الإيجابية والأخرى تطبق الفكرة الجديدة في البحث مع الأخذ بالنموذج المقترح لبروتوكول الانتقال . وتشير نتائج المحاكاة إلى أن الفكرة الجديدة تعطي تحسيناً واضحاً على أداء السرعة العالية .

Key Words: Performance evaluation, High speed protocol, Transport layer, Simulation model, Window's lower and upper bound, Periodic exchange of information.

ABSTRACT

Transport protocols that were designed for network today are not adequate for high speed networks of the future due to the increase in network bandwidth relative to the speeds of the processor components. As a result, 100 Mb/s-1Gb/s high speed networks are now available but however at the top of transport layer, 10 Mb/s or even less is only achievable with current protocols. This increases the demand on new architectures and protocols to be designed and implemented.

In this paper we suggest a model for the transport protocol and implement the idea of exchanging the complete state information between the transmitter and receiver periodically. This idea was introduced by AT&T. We implement two versions of the simulator, the first version implements the positive acknowledgment method, the second version implements the new idea. Both simulators implement the same proposed model for the transport protocol. The simulation results show that the new idea makes a big enhancement on the performance in the high speed environment.

INTRODUCTION

It is clear that the data networks we inherited from the 1980's are inadequate to handle the applications and capabilities required by the 1990's. Today packet switched data networks face a number of problems. They are high cost, low speed, and they introduce large switching delay and relatively high error rates. The switches require too much storage in the network. The protocols are too heavy weight and there are too much processing done in the network.

In this paper we are concerned with high speed transport protocol. This needs a dramatic change in thinking and in

structures to handle the new requirements for high speed networks [1]. We will discuss the results of a set of simulation experiments to study the most suitable implementation for the data transfer phase in the proposed high speed transport protocol. The proposed protocol can be directly combined with any existing connection management protocols to provide both connection management and transfer services [2]. In the proposed protocol we use fixed format packets with all the fields within a packet with fixed places and fixed length. Unlike current transport protocols which handle both flow control and error control through combining sequence number and window technique, we separate these two functions and handle flow control through combining both rate control and window techniques. We also implement the idea of periodically exchange full state information between the transmitter and the receiver which is supported by AT&T Bell lab [3] and compare its performance with the performance of positive acknowledgment method. We try in the design to maximize the functional parallelism which is based on the configuration of the protocol functions inside the protocol entities by keeping the protocol functions as much independent as possible. We suggest four different processes for the receiver and four for the transmitter. We have enhanced the performance of the proposed protocol by this parallel implementation.

PROBLEM DEFINITION AND DIFFERENT APPROACHES

Distributed processing, full motion video, video on demand, and computer imaging are examples of applications which need giga bit per second service for end users [4]. Fiber optic networks which can offer the required high bandwidth have also became available [5, 6]. However the performance of the complete communication system including all the layers of the OSI protocol stack has not significantly increased [7]. The reason is the increasing gap between the physical network bandwidth and the processing power available for protocol processing. The bandwidth of networks is increasing at least one order of magnitude faster than the performance of the processors. The performance bottleneck of modern communication system is thus no longer the network bandwidth but the processing power inside the communication nodes.

Problems With Current Transport Protocols

As network rates reach the 100Mb/s - 1Gb/s range it will be more difficult for processors to process packets in real time that is as fast as they arrive at the network interface. For example a 1024 byte frame arriving at a 56 Kb/s interface, will take 146 ms to be received from the network. This gives the processor 146 ms to process the previously received packet. The same frame arriving at 1 Gb/s interface must be processed in approximately 8 µsec. Failure to process packet in real time will cause buffers at the receiver to fill and eventually lead to lost packet due to receiver overflow.

Approaches to overcome transport bottleneck

To overcome the transport bottleneck in high bandwidth networks two major approaches can be distinguished.

1. The use of novel implementation techniques that lead to more efficient implementation of current protocols [8, 9, 10].

2. The development of light weight protocols that require less processing (high speed protocols) [11, 12, 13, 14, 15].

Figure 1 summarizes the different approaches to overcome the transport bottleneck.

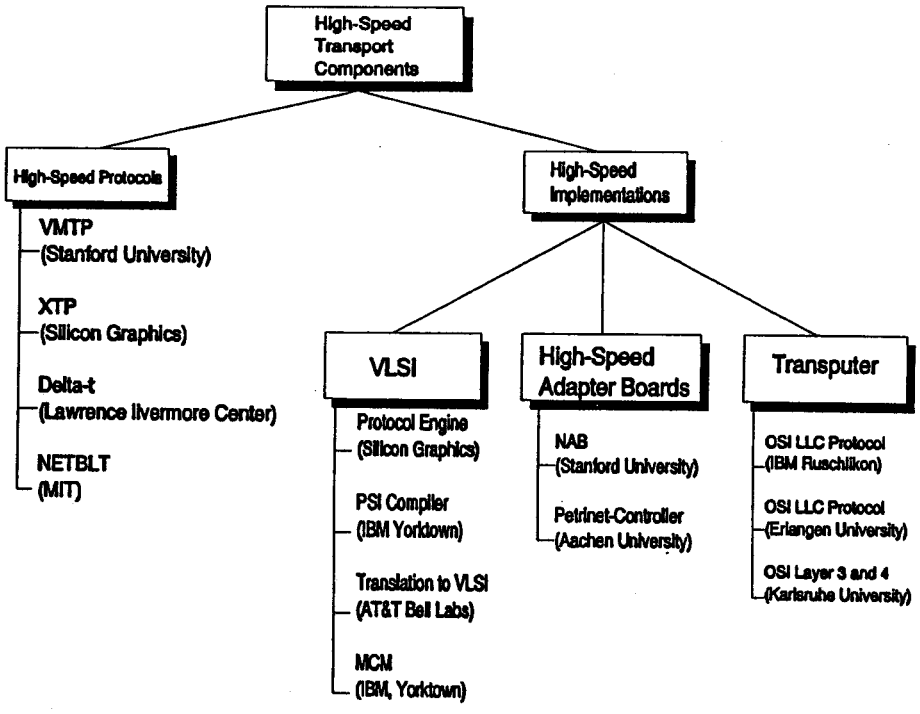

Fig. 1: Different approaches for design of high speed transport protocols.

This protocol is characterized by the following

1. Flow control is implemented through rate and window method.
2. Error Recovery is done by periodic exchange of information, selective retransmission, and fewer times.
3. Connection setup and release is implemented through three way handshake.
4. Implementation is seperate from the host operating system.
5. Packets are of fixed formats.
6. There is no multiplexing.
7. Implementation of the protocol is parallel in a dedicated front end communication processors.
8. Data movement is minimized as data in a packet moves through the bus only once, when the host reads or writes it. The processors on the transmitter and receiver sub buses access the packet headers, but do not read or write the data in the packet.

Now, we will give a brief description of how this protocol works :

At Transmitter

A table Tpacket [serial #, retrans, ack, address] which has a record for each outstanding packet, is maintained and is updated as new state information becomes available from the receiver, where serial # is the serial number of an outstanding packet, retrans is its retransmission count, ack is a bit indicating whether it has been successfully acknowledged or not, and address is the packet address in the buffer.

Retransmission Time Out

The retransmission time out is handled through the variable retrans which is assigned to each outstanding packet in term of the number of received control packets. A packet is scheduled for retransmission only if it is not yet acknowledged and its retransmission count is zero. This mechanism removes the need of explicit timers in the transmitter and gets rid of the problem of retranslating a packet because it is a little bit delayed as when the load on the receiver increases, and hence the effective round-trip time through the system increases, and the retransmission time-out increases automatically.

At Receiver

A table Packet [serial #, address, j] is maintained at the receiver and is updated as new packets arrive from the transmitter. Where j is set to 1 (0) if the corresponding packet is received (not received) correctly.

Control Packet Transmission Time

Both the transmitter and the receiver send their control packets periodically containing the current status for each active connection. Control packet interval=nominal part+time to process sending control packets to all active connection + variable time that increases as the load increases.

This scheme has the following advantages

1. If the receiver is extremely busy, i.e. if there are many packets waiting to be processed or if there are few free packets then the control packet is delayed by increasing the variable part in the control packet interval. This will help in preventing "retransmission avalanche".
2. For low activity connection, some of the control packets will be skipped, i.e. sending one control packet and skip one or two control packets. However the control packet serial number will be still increased as if all the control packets were sent, so the effective re transmission time out period remains the same.

This protocol has two types of packets; control packets, and data packets. The control packets have two types; type (0) which contains receiver's state, and type (1) which contains the transmitter's state. Both types of control packets as well as data packets are shown in Fig. 2, where LCI is the Logical Connection Identifier, serial # is packet serial number, rcv-uw is the receiver window upper bound, trans-uw and trans-lw are the transmitter window upper and lower bounds. The buffer available is used for flow control and bit map is used for acknowledgment. No. of packets queued is used in congestion control to decide whether the receiver should accept another connection or not. Control packet containing the transmitter's state and receiver's state are sequenced independently. Out of sequence control packets are discarded, and control packets with monotonically increasing number are processed.

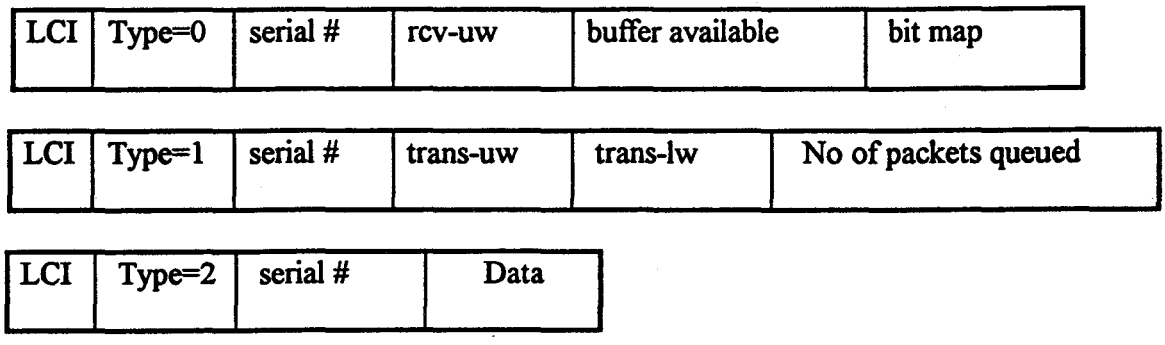

Fig. 2: The different types of control and data packets.

SIMULATION MODEL

To simulate the transport layer we have two options: either to use a global model for representing the effect of the next lower protocols or to use an analytical model of the transient delay which takes into account the stochastic variation introduced by the communication network.

We used the second approach because it is a hybrid model and it provides a good compromise between the difficulties in mathematical modeling and the multiplicity of interacting resources on one hand, and the inclusion of the lower level details in a pure simulation on the other hand. In particular, the hybrid simulation of transport protocols would require to have accurate global models of interesting performance measures of the underlying communication protocol, so that an appreciable amount of low level details could be removed without much sacrificing in the simulation precision.

The network model for transient delay is taken as

$$T = N. [S + W]$$

where, T is packet transient delay from source to destination node switches, N is number of hops between source and destination for the considered packet (geometric distribution), S is link service time corresponding to the packet, and W is a random variable with suitable statistical characteristics which accounts for queuing delay in node switches [16].

The simulation model of the new transport protocol is summarized in Fig. 3. In the simulation, each machine is modeled as a "process" that operates synchronously with each other. A packet generation mechanism fills the host-trans queue which acts as the main source of the traffic. Exponential distribution is used for TPDU interarrivals. Number of parameters such as number of hops, mean inter TPDU interval, underlying network parameters, ...etc., are all user selective.

The simulator is divided into two parts, the transmitter and the receiver. Each part consists of processes, fifo's, and state data.

A. The Transmitter

The transmitter processes maintain the window's lower and upper bound (trans-lw, trans-uw), pointers to window's worth of percolated packet buffers, a copy of the last control packet from the receiver, outstanding packets (i.e. packets in the range trans-lw and trans-lw+L-1), a bit flag, acked which means that: "the packet has been acknowledged by the receiver", and a table Tpacket [serial #, retrans, ack, address] for each connection. The transmitter consists of host-trans, trans, get-rcv-stat, and send-trans-stat processes.

The host-trans is responsible for deciding whether to accept a new packet from the session layer or not. It takes its decision depending on the buffer availability at the receiver and the window's upper and lower bound values. In case of acceptance, it gets the address of the next free packet buffer from the free packet fifo, fills in the data part of the packet buffer and writes the address of the packet to the trans fifo. It also advances the window's upper bound and initializes the table entry for the new packet. When a connection is established, the packet buffers are allocated for the connection. The packet headers are initialized and the packet buffer addresses are placed on the fifo.

The trans process has three input fifos of pointers to packet buffers. The first one with high priority is used for control packets which is emptied first, the second fifo is used

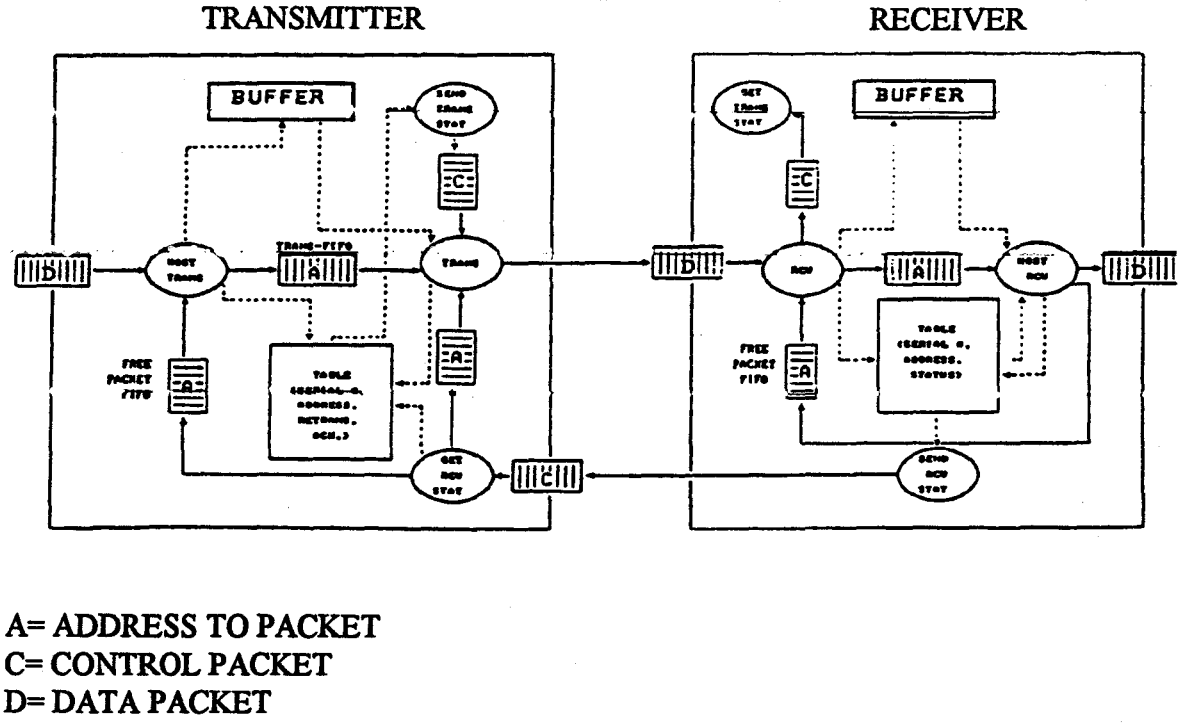

Fig. 3: Proposed Protocol Simulation Model.

for data packets whose restransmission time out expires and they are not acknowledged (retransmitted packets), and the third fifo is used for new data packets. Trans process writes also the retrans entry in the tpacket table for the corresponding packet before sending it to the underlying network.

The get-rcv-stat process takes the receiver control packet and updates the transmitter state. After updating the transmitter state, it tests if any packet need to be transmitted, and puts its address in the retransmitted fifo. It also returns the address of the acknowledged packet to free packet fifo.

The send-trans-stat process sends the transmitter's state periodically to the receiver.

B. The Receiver

The receiver maintains the window's lower and upper bound (rcv-lw, rcv-uw), the current status data (bitmap, control packet serial #, ...), the next packet number to give to the host, and a pointer to the packet buffer for each packet or an indication that this packet has not been received, this is an array indexed by the packet number modulo the window size (packet [i]) for each connection. The receiver consists of rcv, host-rcv, send-rcv-stat, and get-trans-stat processes.

The rcv process receives the packet from the network layer, and it tests for duplicate packet. If it decides to accept the incoming packet, rcv process takes the address of the packet buffer from the free packet fifo and writes the data packet to the buffer and the address to host-rcv fifo. Then it updates the corresponding entry in the packet table to indicate the receiving of this packet.

The host-rcv provides the host with order packets. It tests for the next packet in its fifo, if it is the next packet to be delivered to the host, it delivers it and its subsequent packets if any, and it puts their addresses in the free packet fifo, and advances the window upper bound as much as possible.

The Send-rcv-stat sends periodically the control packet containing the current status for each active connection to the transmitter. It also advances the window lower bound, and initializes the entry in the packet table corresponding to delivered packets.

The Get-trans-stat takes the transmitter control packet and makes use of the available data on any further decision.

SIMULATION RUNS

The performance measures used in the simulation are the throughput and the wasted traffic (overhead). The effect of different parameters is studied in the simulation runs to see the behavior of the protocol with different values for a certain parameter.

We have investigated three packet error rates: one loss per 1000 packets, one loss per 10,000 packets, and one loss per 100,000 packets. For 1 Kbyte packets these packet error rates correspond to bit error rates of 10^{-7}, 10^{-8}, and 10^{-9}. Although this bit error rate may seem a little bit high for optic fibers, we must put in consideration that packet error rate includes also losses in packet switches due to congestion.

We have investigated one way network transmission delays of 5, 10, 15, and 20 ms/packets which correspond to a distance range from 1500 km to 6000 km which is a very adequate range to study in WAN.

The protocol itself has a number of parameters. The first parameter is the window size which we have studied its effect on throughput. We choose a window size of 64, 128, 256, and 1024 packets. We have not increased the window size more than 1024 packets because the transmitter and the receiver must reserve one window's worth of packet buffer for each active connection. So the large window size requires large amount of buffer memory at the receiver. We arbitrarily set the maximum window size to 1024 packets which corresponds to 1 Mbyte of buffer dedicated to each connection.

Another important parameter studied which has effect on both the throughput and the wasted traffic is the re transmission time out. For each network delay we have studied the effect of changing the retransmission time out.

We have tried varying the interval between information exchange while holding the time out constant and see the effect of period of information exchange.

Another very important parameter which is related directly to the implementation of the used protocol H/W and S/W, is the processing time. We have taken a range from 60μs to 130μs for processing one packet.

RESULTS & CONCLUSIONS

In general it is very obvious that the periodic exchange of information in all our runs gives better throughput and lower overhead. This is expected because the periodic exchange of information method removes many of the recovery procedures required to overcome loss of control packets, because even if there is an error in the transmitted control packet, the next packet does not send incremental information from the previous one but sends complete information which corrects it and therefore all control packets received in error are simply dropped. In positive acknowledgment method if the acknowledgment is lost or delayed the packet is retransmitted once more after waiting for its retransmission timer to expire. These are the general reasons for having better performance.

It is not only that the new method has better performance results but it also has some interesting properties as we will discuss with the related figures. In Fig. 4 we see that the throughput increases rapidly with decreasing the processing time but it almost stops at 0.08 in the new method which corresponds to packet inter arrival time, while it continues increasing in the normal acknowledgment method. Fig. 5 shows that increasing the window size has greater effect on the new method than on the old one. Moreover, the new method begins to stabilize at window size 1024 packets while the old method does not. A very important draw back for increasing the window size in the old method is increasing the overhead, while this drawback does not

appear in the new method. The effect of the retransmission time out in both methods is shown in Fig. 6 through 9. Another very interesting result is that the new protocol is less sensitive to the retransmission time out and there exists a big range retransmission time out where we can get the maximum throughput, unlike the old method where the maximum throughput is obtained in one point only. This solves the very old problem of determining the retransmission time out value and makes it no longer a critical point. The previous conclusion is valid for network delay of 5, 10, and 15 ms. For network delay of 20 ms both methods show approximately the same sensitivity. This is because in case of 20 ms delay a window size of 1024 is not capable of making the network pipe full and waiting for retransmission timer to expire in case of error costs a lot.

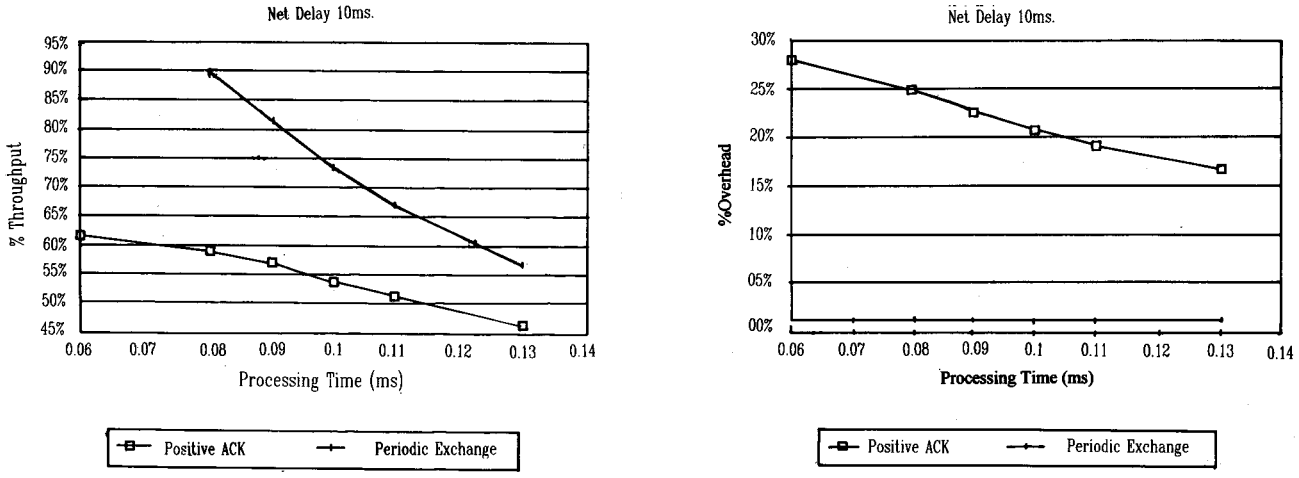

Fig. 4: Effect of varying packet processing time on throughput and on overhead.

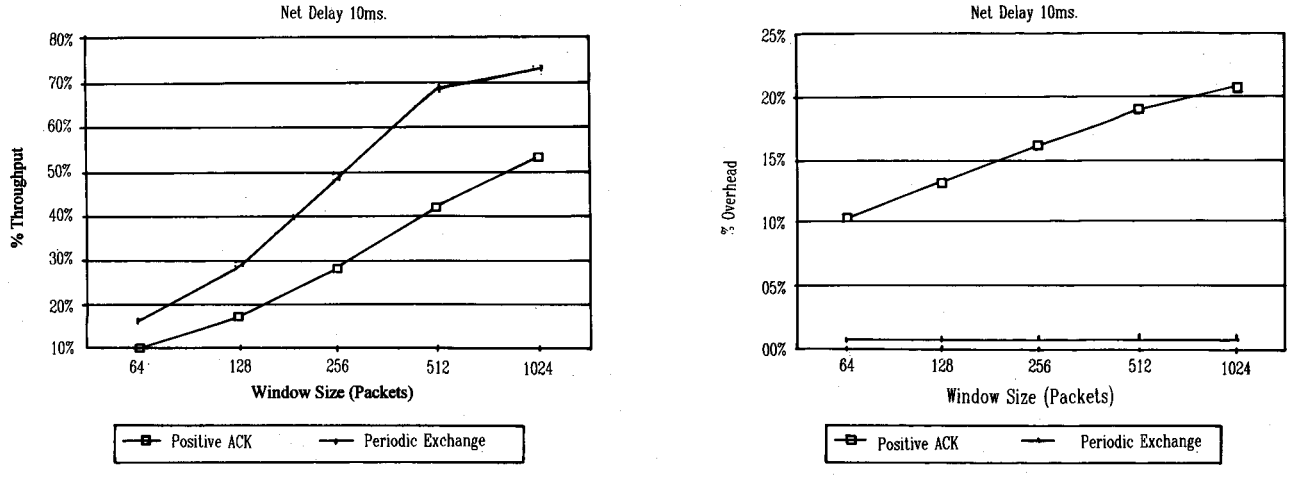

Fig. 5: Effect of varying window size on throughput and on overhead.

packet error rate=0.001 period of information exchange=5ms retransmission time out=30ms.

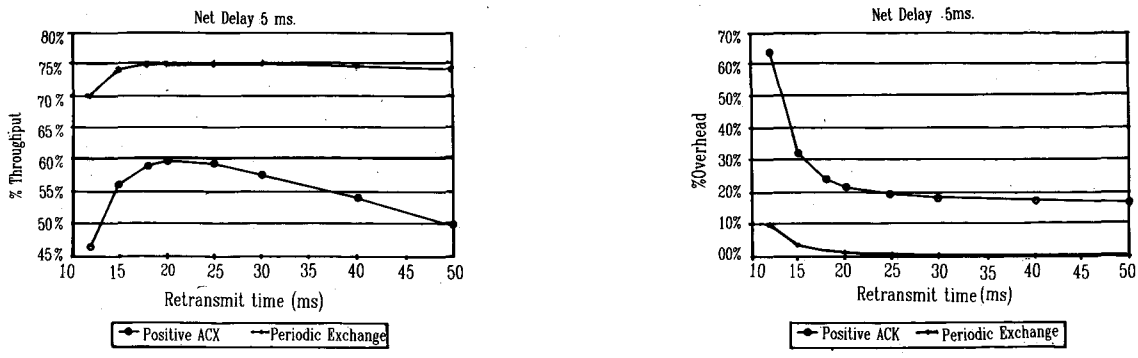

Fig. 6: Effect of varying retransmission time out for network delay 5ms.

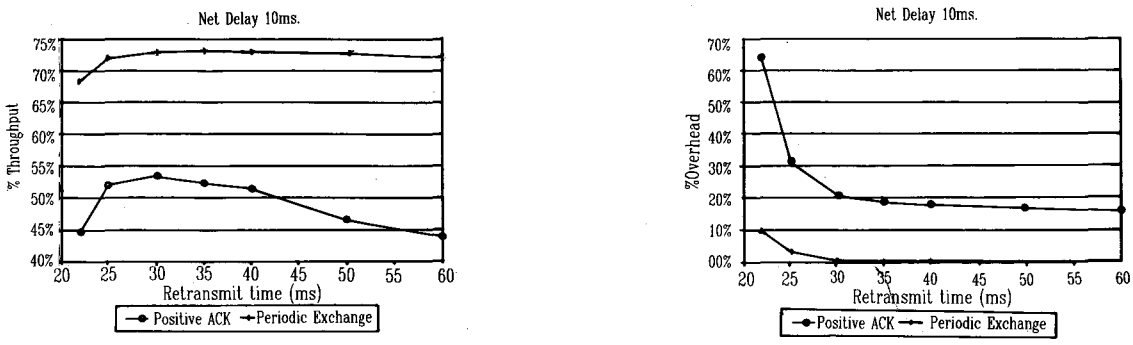

Fig. 7: Effect of varying retransmission time out for network delay 10 ms.

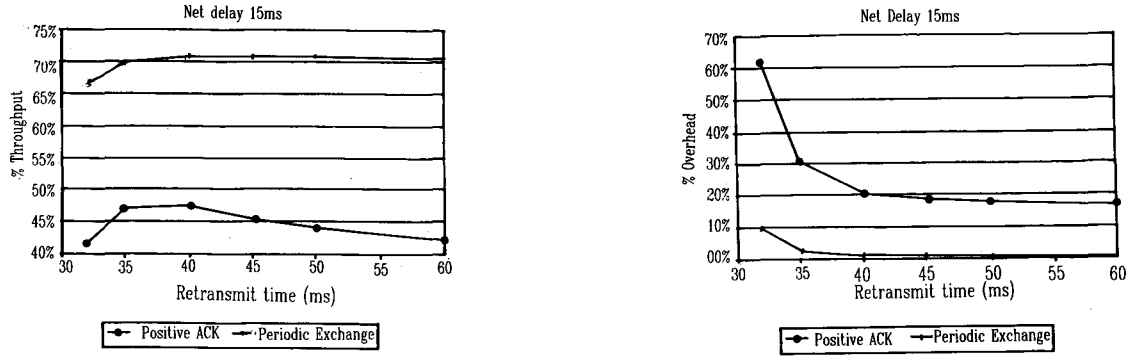

Fig. 8: Effect of varying retransmission time out for network delay 15ms.

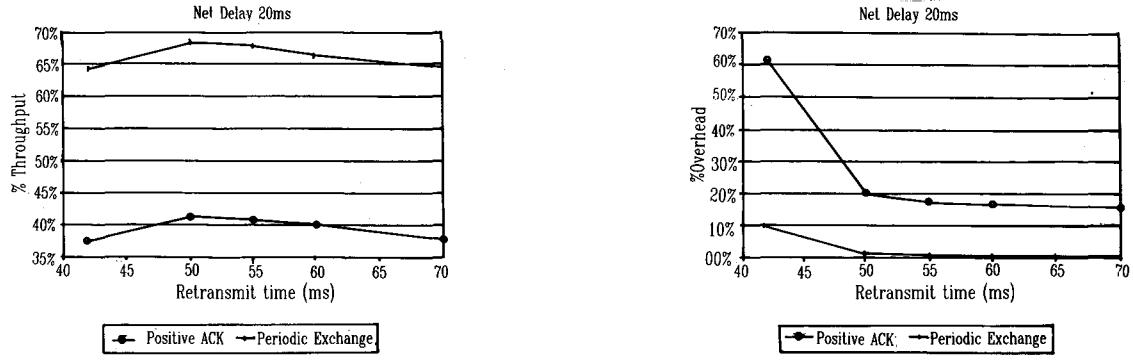

Fig. 9: Effect of varying retransmission time out for network delay 20ms.

REFERENCES

[1] **Hac, A.** and **Hasan B. Multu, 1989.** Synchronous optical networks and broadband ISDN protocols, IEEE Computer, No. 11.

[2] **Murphy, S.L.** and **A. Udaya Shankar, 1991.** Connection management for the transport layer: service specification and protocol verification, IEEE transactions on communication, vol. 39, No. 12.

[3] **Netravali, A., W. Roome** and **K. Sabnani, 1990.** Design and Implementation of a high speed transport protocol, IEEE Trans. on comm. vol 38, No. 11.

[4] **Wright, D.** and **Michael To, 1991.** Telecommunicaton Application of the 1990s and their transport requirements, IEEE Network magazine, No. 3.

[5] **Kleinrock, Leonard, 1991. ISDN**: The path to broadband networks, Proceedings of the IEEE, vol. 79, No. 2.

[6] **Joshi, Sunil P., 1986.** High-performance networks: A focus on the Fiber Distributed Data Interface (FDDI) standard, IEEE Micro,No. 6.

[7] **Meister, B., 1991.** A performance study of ISO Transport Protocol, IEEE Trans. on computers vol. 40 No. 3.

[8] **Zitterbart, M., 1991.** High Speed Transport Component, IEEE Network magazine, No. 1:54-63.

[9] **Davie, Burce S., 1991.** A host-network interface architecture for ATM, ACM SIGCOMM' 91 conference on communication architectures and protocols, Zurich, Switzerland.

[10] **Brendan C., S. Traw** and **Jonathan M. Smith, 1991.** A high performance host interface for ATM networks, ACM SIGCOMM' 91 conference on communication architectures and protocols, Zurich, Switzerland.

[11] **Zitterbart, Martina** and **Ahmed N. Tantawy, 1991.** Transport Service and protocols for high speed networks, IBM research report.

[12] **Doeringer, W., D. Dybeman, M. Kaiserwerth, B. Meister, H. Rudin** and **R. Williamson, 1990.** A Survey of light-weight transport protocols for high speed network, IEEE Trans. on comm. vol. 38 No. 11.

[13] **Laporta, T.** and **M. Schwartz, 1991.** Architectures, Features and Implementation of high speed transport protocols, IEEE Network magazine, No. 5:14-22.

[14] **Koufopavlou, O. G., A. N. Tantawy** and **M. Zitterbart, 1992.** Analysis of TCP/IP for high performance parallel implementations, proceedings of the 17th conference on Local Computer networks, Minneapolis, Minnesota:576-585.

[15] **Tantawy, A. N.** and **M. Zitterbart, 1993.** Multiprocessing in high performance IP Routers, Protocols for high-speed networks, IFIP, Elsevier Science publishers B.V., North Holland.

[16] **Saber, Amal, 1990.** Performance evaluation of transport layer protocols M.Sc. thesis, Faculty of Engineering, Alexandria University.

[17] **Green, Paul E., 1991.** The future of fiber optic computer networks, IEEE Computer, No. 9.

[18] **Law, Averill M.** and **W. David Kelton, 1982.** Simulation modelling and analysis, McGraw-Hill.