ISSN: 0128-7680
Pertanika J. Sci. & Technol. 13(2): 187 - 211 (2005)

A New Congestion Control Algorithm for Active Networks

[1]M.I. Buhari, [2]M.H. Habaebi & [2]B.M. Ali
[1]King Fahd University of Petroleum and Minerals,
Dhahran, Saudi Arabia
[2]Department of Computer and Communications Engineering
Universiti Putra Malaysia
Serdang 43400 Selangor, Malaysia

Received: 9 April 2003

ABSTRAK

Makalah ini membincangkan dan menganalisis skim kawalan kesesakan yang berbeza untuk rangkaian-rangkaian aktif dan mencadangkan rangkaian baru. Skim yang dicadangkan mengenai satu keseimbangan antara bilangan kesilapan yang dipertimbangkan dan masa menunggu sebelum mengambil tindakan. Skim yang dilaksanakan bertindak ke atas konsep "Choke packet", iaitu sumber tersebut diarahkan untuk mengurangkan kadar transmisi. Pengurangan kadar transmisi yang munasabah dikekalkan pada sumber tersebut sambil nod perantaraan dalam pelantar jaringan aktif bertindak sebagai ganti biasanya nod matlamat bertindak. Skim yang dicadangkan disimulasi menggunakan OPNET dan keputusan simulasi dilaporkan. Keputusan menunjukkan bahawa teknik pelindapan sumber dilaksanakan dalam nod perantaraan jaringan aktif mengurangkan kesesakan secara drastik dan memperbaiki keseluruhan prestasi jaringan.

ABSTRACT

This paper discusses and analyses different congestion control schemes for active networks and proposes a new one. The proposed scheme strikes a balance between the number of errors considered and the waiting time before taking action. The scheme implemented works on "*Choke packet*" concept, where the source is instructed to reduce the transmission rate. Reasonable reduction of transmission rate is maintained at the source as the intermediate nodes in active networks platform take actions instead of the usually acting destination nodes. The proposed scheme was simulated using OPNET and simulation results are reported. Results indicate that the source-quenching technique implemented in the intermediate nodes of the active network reduces congestion drastically and improves overall network performance.

Keywords: Congestion control, source quench, choke packet, active network, OPNET

INTRODUCTION

The goal of congestion control is to control input traffic flows, so as to provide a reasonable tradeoff between packet losses and overall network throughput. Recently, many research findings have been reported in the area of congestion control in active networks. Rejaie *et al.* (1999) discussed the use of Rate Adaption Protocol (RAP) that suffers from the drawback of changing the rate of transmission at the source to half in case of congestion. Sisalem and Wolisz

(2000) discussed the Loss-Delay adaptation algorithm that uses Real-time Control Protocol (RTCP) for feedback messages. This algorithm too suffers from the slow response of RTCP that makes the congestion control scheme not fast enough to respond to the dynamics of active networks. TCP-Friendly Rate Control Protocol (TFRC) described by Floyd *et al.* (2000) starts slowly but initially takes actions on stabilizing the network when a single loss occurs. A combination of Window-based and Rate-based congestion control algorithms was proposed by (Rhee *et al.* 2000) where action is taken after some fixed number of errors has occurred instead of at every occurrence of error.

As the amount of processing increases due to parallel packet filtering concept being implemented in active networks, the probability of having congestion also increases. The fact is that the parallelisation of the access control rules increases the number of rules needed to represent the access list. To handle congestion, various congestion control schemes were analysed and discussed and a new scheme was proposed in this paper. The proposed scheme strikes a balance between number of errors considered and the time required to handle errors or congestion. The scheme implemented works on the "*Choke packet*" concept to handle congestion, where the source is instructed to reduce the transmission rate whenever there is congestion due to huge traffic. Reasonable reduction of transmission rate is maintained at the source as the intermediate nodes in an active networks platform take action instead of the usually acting destination nodes.

Related Work

Congestion control schemes can be of two types, namely, Window-Based & Rate-Based. Window-based category uses a congestion window either at the sender or the receiver end. Similar to TCP, each packet transmitted consumes one slot in the congestion window, while each packet received or the acknowledgment of a packet received frees one slot. The sender is allowed to transmit packets only when a free slot is available.

Rate-based category adjusts the transmission rate dynamically according to some network feedback mechanism. It can be subdivided into simple AIMD (Additive Increase, Multiplicative Decrease) schemes and model-based congestion control. Simple AIMD schemes mimic the behaviour of TCP congestion control. This results in a rate that displays the typical short-term saw-tooth like behaviour of TCP. This makes simple AIMD schemes unsuitable for continuous media streams. Model-based congestion control uses a TCP model by adapting the sending rate to the average long-term throughput of TCP. Model-based congestion control can produce much smoother rate changes that are better suited for continuous traffic (Widmer *et al.* 2001).

As in the case of Packet filtering, the usage of different access lists can vary the processing and in turn the impact of congestion on the network varies. Thus, we use the rate-based congestion control scheme. This helps us to change the rate dynamically when there is some change of access lists and so on. As the traffic is not continuous, we choose to use the simple AIMD form of the Rate-based category.

The following parameters are considered to compare the performance of congestion control scheme:

1. Decision Function: This indicates when the change in transmission rate is made. What is the impact of the protocol with and without congestion existence?
2. Decision Frequency: How frequently the changes are made. Do these changes impact the total performance of the network?
3. Stability: After a network undergoes some perturbation because of flows joining and leaving, if the network reaches steady state, no matter what the state of the protocol at the end of the perturbation is, the protocol eventually reaches the fair and TCP-friendly rate.
4. Scalability: The performance of a protocol for instance does not depend on the number of its receivers.

Various Congestion Control Schemes:

We have considered many rate based congestion control algorithms. The concepts behind those congestion control schemes and the impact of decision function and decision frequency on these congestion control schemes are discussed below.

1. The Rate Adaptation Protocol (RAP) (Rejaaie *et al.* 1999) is a simple AIMD scheme for unicast flows. The receiver acknowledges each data packet. The ACKs are used to detect packet loss and infer the round-trip time. When the protocol experiences congestion it halves the sending rate. In periods without congestion, the sending rate increases by one packet per round-trip time, thus mimicking the AIMD behaviour of TCP. The decisions on rate increase or decrease are made once per round-trip time. While TCP congestion control is appropriate for applications such as bulk transfer, some real-time applications (that is, where the data is being played out in real-time) find halving the sending rate in response to a single congestion indication to be unnecessarily severe, as it can noticeably reduce the user-perceived quality (Tanenbaum 1996). Thus, the impact of reducing the data transfer abruptly disturbs the network.

   With regard to the decision frequency, changes are made every round-trip time (RTT). Due to the random nature of RTT, using the recent sample RTT as the step length is likely to result in a poor behaviour. RAP connections with shorter RTT are more aggressive and achieve a larger shareof the bottleneck bandwidth. However, there are two issues to notice. First, in general, other measures of fairness can only be achieved by implementing the required ~Gchinery in the network. Second, as long as the unfairness problem is not resolved along TCP flows, being TCP-friendly implies accepting this unfairness.

2. Loss-Delay Adaptation Algorithm (Sisalem and Wolisz 2000) does not devise its own feedback mechanism to control the sending rate but relies

solely on the Real-Time Transport Control Protocol (RTCP) feedback messages provided by the Real-Time Transport Protocol (RTP). The increase and decrease factors for AIMD are dynamically adjusted to the network conditions. The amount of additive increase is then determined as the minimum of three independent increase factors to ensure that (i) flows with a low bandwidth can increase their rate faster than flows with a higher bandwidth; (ii) flows do not exceed the estimated bottleneck bandwidth; and (iii) flows do not increase their bandwidth faster than a TCP connection. The authors claim that the slow rate increase of LDA+ compensates for the fact that the rate is at most decreased to the throughput estimate of the complex TCP model. However, RTCP reports are generated infrequently (usually within several seconds). This makes LDA+ slow to react to changes in the network conditions. While looking at the decision function, we infer the following:

- LDA+ uses the real-time transport protocol (RTP) for collecting loss and delay statistics that are then used for adjusting the transmission rates of the senders in a manner similar to TCP connections suffering from equal losses and delays.
- With no observed losses, the sender can increase its transmission rate additively, otherwise it needs to reduce it multiplicatively. For the case of no losses, the flow's share can be increased by a value that does not exceed the increase of the bandwidth share of a TCP connection with the same round trip delay and packet size.

  The in-frequency of the RTCP feedback messages dictates that an RTP sender cannot benefit fast enough from rapid changes in the network conditions. Thus, the goal of RTCP-based adaptation is to adjust the sender's transmission rate to the average available bandwidth and not react to rapid changes in buffer lengths of the routers.

3. TCP-Friendly Rate Control Protocol (TFRC) (Floyd *et al.* 2000) starts with a slow start phase but stops the phase at first loss event. The sender then computes a new fair rate and changes the sending rate. Several requirements for a loss rate estimator are formulated and the authors settle on the Average Loss Interval method which best fulfills these requirements. Once per round-trip time the TFRC receiver updates its parameters and sends a state report to the sender. The sender then computes a new fair rate from these parameters and adjusts the sending rate accordingly. A major advantage of TFRC is that it has a relatively stable sending rate while still providing sufficient responsiveness to competing traffic. As in Packet filtering, the network setup is made with proper care on the access lists and the access lists do not change often, it is not necessary to change the sending rate often. While looking at the decision function, we infer the following:
   - The sender explicitly adjusts its sending rate as a function of the measured rate of loss events, where a *loss event* consists of one or more packets dropped within a single round-trip time. Loss event rate is

different from loss packet rate because the loss impact on successive packets is counted as an event.

- In response to a decrease in the loss event rate, the sending rate is increased slowly. Similarly, do not halve the sending rate in response to a single loss event. However, in case of successive loss events, halve the sending rate.

With regard to the decision frequency, the receiver should report feedback to the sender at least once per round-trip time if it has received packets in that interval. If the sender has not received feedback after several round-trip times, then the sender should reduce its sending rate, and ultimately stop sending altogether.

4. TCP Emulation At Receivers (TEAR) (Rhee *et al.* 2000) is a hybrid protocol that combines aspects of window-based and rate-based congestion control. TEAR receivers calculate a fair receive rate which is sent back to the sender, who then adjusts the sending rate. To this end, the receivers maintain a congestion window that is modified similarly to TCP's congestion window. In contrast to TCP, the TEAR protocol does not directly use the congestion window to determine the amount of data to send but calculates the corresponding TCP sending rate. While looking at the decision function, we infer the following:
   - If there is no congestion at slow-start or congestion-avoidance state, then the number of packets on transit from source to destination (congestion window size or *cwnd*) is increased by 1/Last *cwnd*.
   - Upon loss of packets, the *cwnd* is not modified. The reception of three packets after the losses will trigger three duplicate acknowledgments in TCP (assuming no delayed acknowledgments).

   With regard to the decision frequency, changes are made when there is a loss of packet. The receiver takes the actions here.

5. Generic router assist (GRA) (Widmer *et al.* 2001), for instance, is a recent initiative that proposes general mechanisms located at routers to assist transport control protocols, which would greatly ease the design and implementation of congestion control protocols.

From the above discussion, it is clear that the change in the transmission rate must not be as often as in RAP or as slow as in LDA+ protocol. The stability of the active network should not be affected by having the processing at the intermediate nodes. To implement congestion control in active networks platform, we incorporated the following steps:

i. In the initial stage of the network, the network starts with a slow-start Algorithm. The slow-start stage is similar to that of the TEAR algorithm.
ii. The intermediate processing node (router) realises late arrival of the packets or out-of-sequence order of packets and then enters a stage of no more increase in the CWND. In case the situation of loss continues then

the intermediate node sends a Source Quench (SQ) message to the source indicating to it to reduce the rate of transmission.

EXPERIMENTAL SETUP

Slow-start Algorithm

When a TCP connection starts up, the TCP specification requires the connection to be conservative and assume that the available bandwidth to the receiver is small. TCP is supposed to use an algorithm called slow start to probe the path to learn how much bandwidth is available (Partridge and Shepard 1977; Lakshman *et al.* 1977). A simple description of the slow-start algorithm is as follows:

(i) Send a TCP packet with one data segment and wait for an acknowledgement.
(ii) Upon receiving the acknowledgement, send 50% more data every round trip.
(iii) Do step 2, until a packet is lost. If so, stop the increase in data size.

Algorithm Design

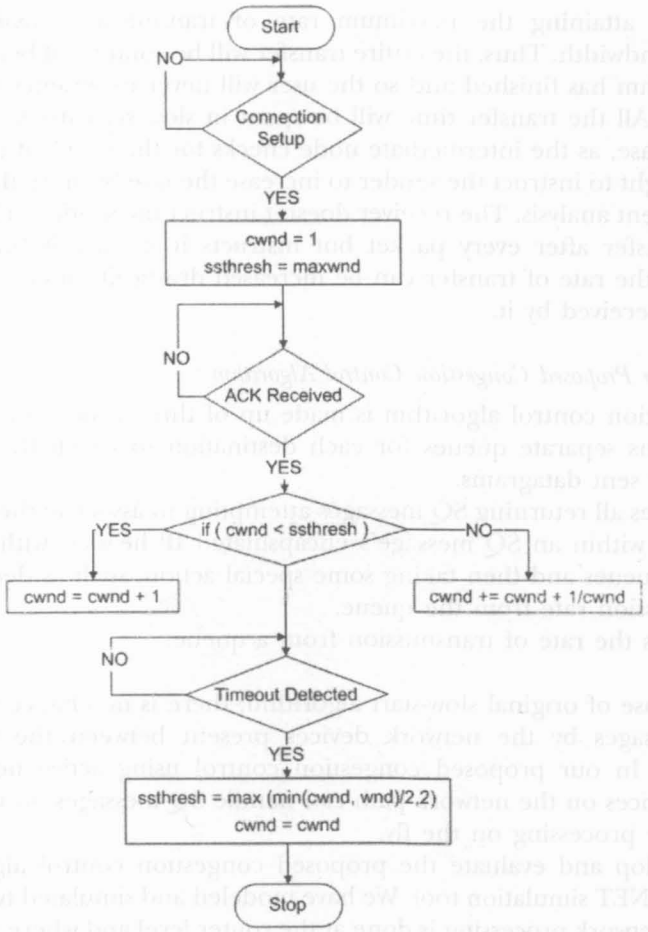
In the case of Slow-start, a back-check packet of full-window size is sent by the sender before it decides about the time-out. The congestion window (*cwnd*) is initially set to 1 and the incrementing is done exponentially. According to Wang and Crowcoft (1992), the congestion avoidance algorithm in fact is a time-out based multiplicative decrease/additive increase algorithm. When a timeout is detected, the threshold *ssthresh* is reduced to half of the current window size and then *cwnd* is stopped from increasing further.

During the slow-start phase, each acknowledgement increases *cwnd* by 1. After *cwnd* reaches *ssthresh, cwnd* is increased by *1/cwnd* for each acknowledgement received (Wang and Crawcoft 1992). The following flowchart summarises the algorithm.

The actual sending window size (*wnd*) would not exceed the receiving size advertised by the receiver (*maxwnd*), therefore the sender always sets

wnd = min(cwnd, maxwnd);

Networks should implement a feedback mechanism that allows traffic sources to control their transmission rate. Within the TCP/IP protocol suite, the ICMP Source Quench (SQ) message is created to provide the feedback from any overloaded gateway or host, back to the source. An ICMP source quench message is intended as a congestion control mechanism in IP. Source quench messages are used when a network gateway cannot forward a message because its message buffers are full. The gateway transmits a source quench message back to the source host machine to request that the source reduces its transmission rate until it no longer receives source quench messages from the gateway. Thus, this effectively throttles back the source's transmission rate. The gateway can transmit multiple source quench messages, one for each packet it receives from a source. The source machine is not required to respond to these source quench messages. When a source receives an SQ message, in principle it should decrease its rate of transmission. In practice, this does not happen (Finn 1995).



When the SQ message arrives, the rate of transmission is decreased. After some time elapses, the source could return to a higher rate of transmission. However, it is safe to keep it lower than the rate that it was using when it first received the SQ message and then slowly allow it to rise. Two phases are present in the case of the Congestion Control algorithm. They are the slow-start phase and the additive phase. In the slow-start phase, the rate of transmission (r) is reduced to a lower value. Then the rate r is allowed to grow. This phase concludes when the rate attains a threshold or another SQ message causes rate r to be reset. The additive phase commences after the rate r reaches the threshold. In this phase, the value of r grows but much more slowly.

Remedy for the Slow-start State Disadvantage

With slow-start algorithm, the rate of transmission is initialised to a slower value and then the rate of transmission is allowed to increase based on response from the destination. There is always a possibility that data transfer would complete

well before attaining the maximum rate of transmission possible on the available bandwidth. Thus, the entire transfer will be completed before the slow start algorithm has finished and so the user will never experience the full link bandwidth. All the transfer time will be spent in slow start itself.

In our case, as the intermediate node checks for the status of the network, it has the right to instruct the sender to increase the rate by more than one step based on event analysis. The receiver doesn't instruct the sender to increase the rate of transfer after every packet but instructs it in case it finds that the increase in the rate of transfer can be increased drastically after a few packets have been received by it.

Procedures for Proposed Congestion Control Algorithm

The congestion control algorithm is made up of three procedures as follows:

i. Maintains separate queues for each destination to which the source has recently sent datagrams.
ii. Examines all returning SQ messages attempting to associate the destination address within an SQ message's encapsulated IP header, with one of the above queues and then taking some special action, such as decreasing the transmission rate from the queue.
iii. Controls the rate of transmission from a queue.

In the case of original slow-start algorithm, there is no chance of handling of SQ messages by the network devices present between the source and destination. In our proposed congestion control using active networks, the network devices on the network path can handle SQ messages, so we handle it using queue processing on the fly.

To develop and evaluate the proposed congestion control algorithm, we used the OPNET simulation tool. We have modeled and simulated two scenarios where the network processing is done at the router level and where the network processing is done at the host level.

Simulation Setup

The set-up consists of a 16-node network designed and simulated using OPNET. Our network model shown in *Fig. 1* consists of 4 subnets each comprising 4 nodes. We made use of OPNET's random packet generation algorithm at each of the nodes. We implemented a static routing algorithm that fixes the next router through which packets can be forwarded from the current one based upon the destination subnet and the host address present in the packet. Secondary routes are assigned for various routes.

As an example, let us suppose that the result of some computation is to be transmitted from node_0 to node_15. The data can pass either through hubs [0, 2 and 3] or [0,1 and 3] (*Fig. 1*). Parallel virtual machine (PVM) resides in active routers (hubs) and also in all the nodes. When data is passing through the first router, the router will partition the computation to be performed on the data into many modules based on availability of nodes. Each module along

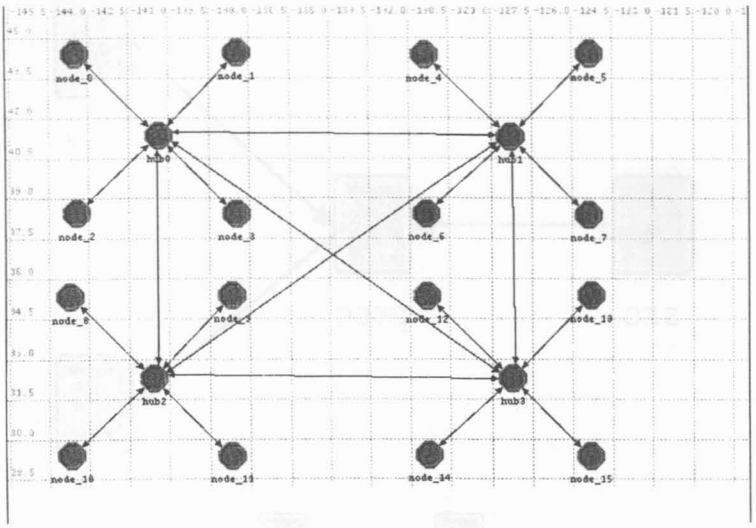


Fig. 1: Our network architecture

with its associated data is sent to one PVM process, which is virtually connected and running in parallel at different nodes connected to the router. The nodes will process the data and send the result to the router. Then the router will forward the accumulated results to the next router, which also forwards the result to the destination [node_15]. The node part of the host and the hub are shown in *Fig. 2*. Congestion is controlled in the router itself, since the router is active and it can run congestion control algorithms.

From *Fig. 2*, we can infer the format of the host and the hub. The hub is made up of a constant packet generator that generates packets at uniform speed. The packet generation is handled by the hosts, which posses the ideal packet generators (**src** in *Fig. 2*) in them. The processing is done on the hub and the packet is sent via the transmitter (**xmt**). Packets enter the hub through the receiver (**rcv**) node present. In the case of the node model of the hub, there exists a receiver and a transmitter for each connection from/to the hub. These connections are called streams. As each hub is connected to three other hubs and four nodes, the node diagram shows seven transmitters and seven receivers. A pair (transmitter and receiver) is connected together in logical form that is shown by the dotted lines.

The packet format for our capsule oriented active networks system is shown in *Fig. 3* and is made up of the following:

The source, destination addresses and the protocol are used for the access control permit or denial decision on the packet. The packet type is used to identify whether the packet is a data packet or a control packet. The timestamp is used to calculate the time taken by the packet to travel from the source to destination. The file name or result and data fields respectively contain the name of the file or the result of access control processing and the data carried by the capsule.

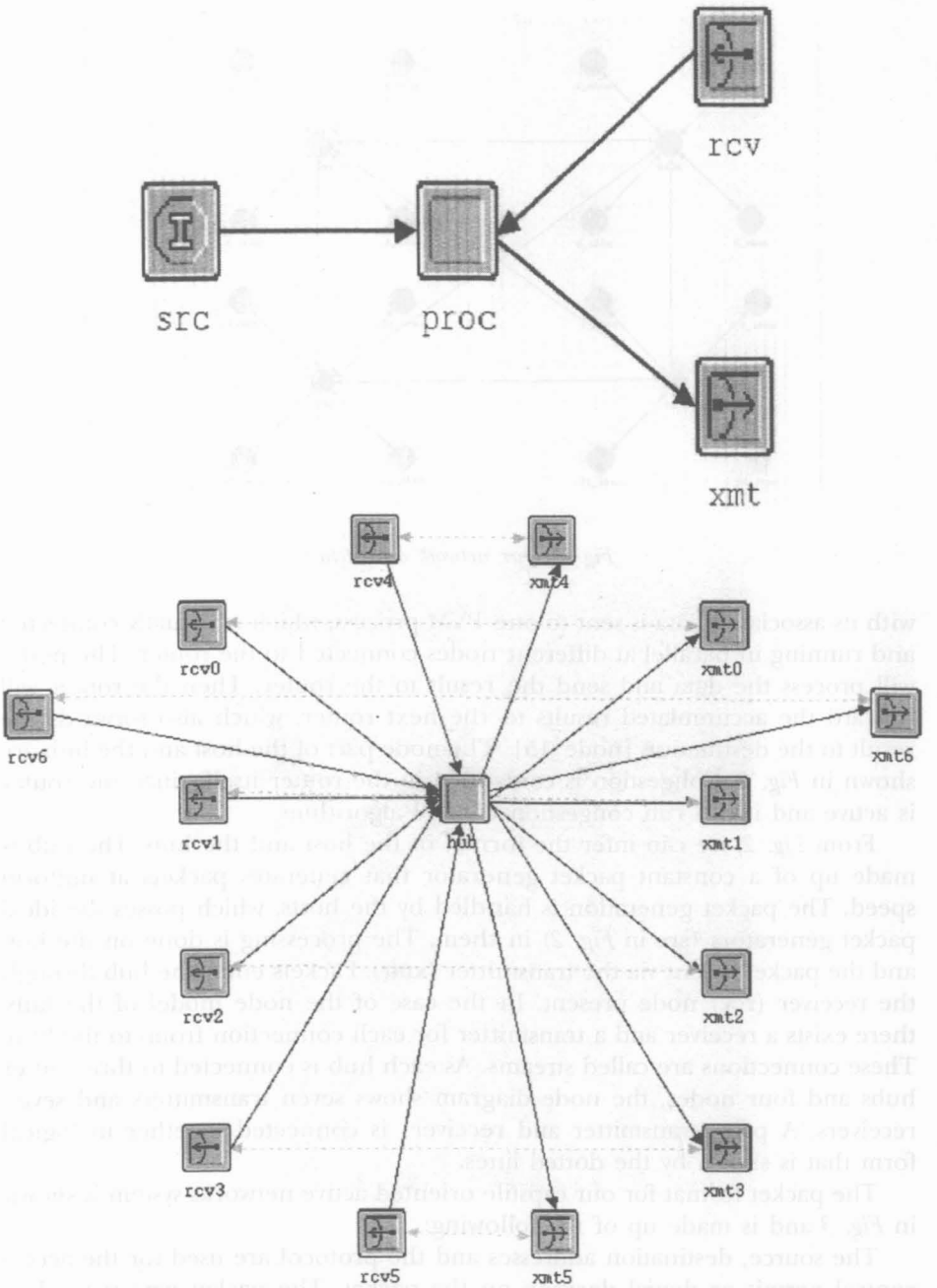


Fig. 2: Node diagram of the host and the hub

Source Address	Destination Address	Protocol	Packet Type	Timestamp	File Name OR Result	Data

Figure 3. Packet Format

The packet format is made up of a source address, a destination address and the type of packet being transmitted. *Type 0* is the normal packet that travels from one node to another. In case of heavy traffic or when the packet arrival rate is more than the capability of the receiver, the receiver sends a SQ packet that is of *Type 1*. Upon receiving the SQ packet, the source reduces the pace at which it sends packets.

Processing Scheme of Host and Hub

The processing scheme of the host and the hub are shown in *Fig. 4*. The host's "**init**" part is used to make the packet destination address generation to fall between 0 and 15. The generated packets are then sent to the idle part. The transmitter then transmits the packet along any one transmit stream depending upon the destination address specified. In *Fig. 4*, the SRC_ARRIVAL indicates the activation of packet transfer from the idle state to the transmitter, upon arrival of a packet that needs to be transmitted. Moments after transmission of the packet, the state comes back to the idle state and waits for transmission or receiving of packets. The '**xmt**' state is the state where the processing that ought to be done before and during the transmission of a packet is specified. On receiving a packet, the RCV_ARRIVAL is activated and the packet is processed by the routines specified in the '**rcv**' state.

At the hub, the checking for the destination address is done by the routines specified in the 'route_pk' state and the packet is forwarded in the respective transmitter stream. The activation of the 'route_pk' state is done by the PK_ARRIVAL activation link. The processing at the hub plays a role in the calculation of RTT due to the fact that there is some active process at the hub that consumes some time. Once the packet reaches the destination, the packet is now processed to identify the time it has taken to reach the destination from the source. Depending upon the time taken an SQ message is generated. If the delay in transmission is for the first time, then the transmission rate is ignored. But if the delay continues then the SQ message is sent to the source.

To identify network congestion, we normally calculate the time taken for the packet to move from the source to the destination. Instead, using active networks, we are concerned with the next intermediate router between the source and the destination. If it takes a long time (which can be determined based on the distance and bandwidth between the node and next intermediate router or hop), it indicates that the network is congested and there is more likelihood of collision and loss of packets. The time calculation between the source and destination not only includes the time taken to travel from source to destination but also the processing that occurs at the hub. To handle this case, the calculations of time at the hubs before the receiving and sending of

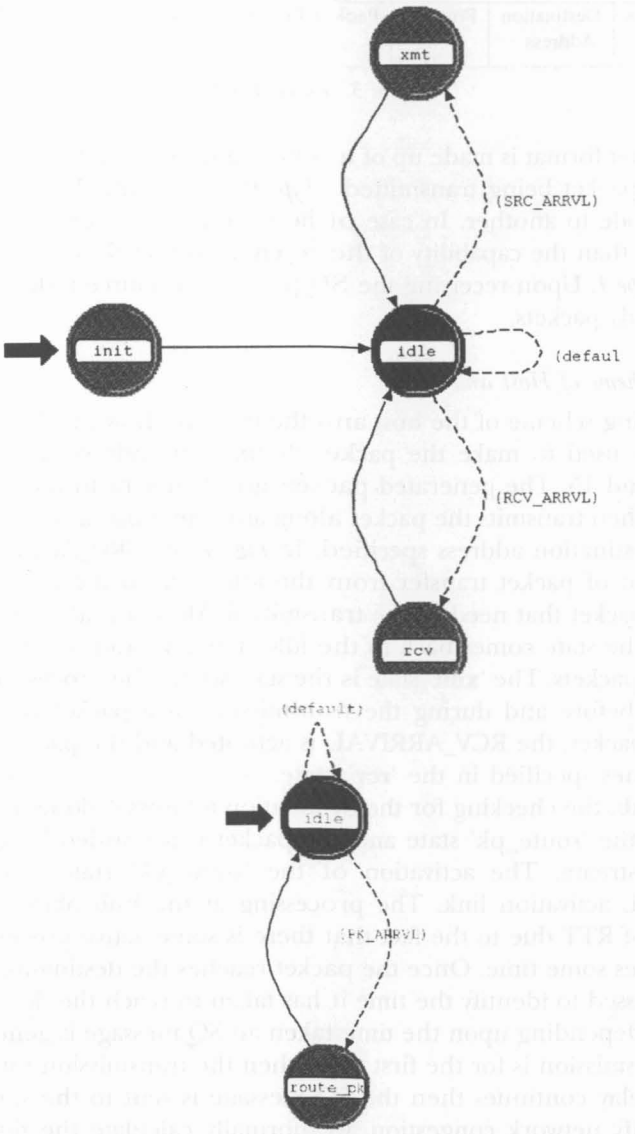


Fig. 4: Process module of the host (top diagram) and the hub (bottom diagram)

the packet will help to calculate the processing time (IP packet filtering time) and in turn can calculate the travel time alone. Based on RTT timing, the source needs to be informed if the destination identifies that the network is congested. After receiving the packet at the destination, the first task done is to calculate the time taken for the packet to reach the destination from the source. The source time is present in the packet header. This time is subtracted

from the current time at arrival to obtain the time taken. The SQ message is generated if the packets are generated at a faster rate by the source continuously. In this case, if the packets are received faster than the receiver can cope with, the receiver will send an SQ message to the source and so the source has to reduce the speed of transmitting packets.

Initial analysis was done for changing the time at which SQ messages are generated along with varying bits per second. Later, the processing was done at two levels, at the end stations and at the intermediate routers, and compared in terms of the recorded performances. In order to see that the situation of applying the active networks concepts of processing the data on the fly helps the congestion control algorithm, we have done the processing at the host level and also at the hub level. The processing time was analysed and the results are reported.

Procedure of Operation for Proposed Scheme

The pseudo code of the different operation procedures of the proposed congestion control algorithm is explained now.

INIT STATE:
Steps:
1. Set the packet distribution to be arranged for a uniform packet distribution.
2. Initialise the source quench, packet count and other variables.
3. Set the packet count value in the packet header.

SENDING NODE:
Steps:
1. Receive the packet from the source stream.
2. Set the destination address in the packet using a random function and then set the source address.
3. The packets are passed along the respective output streams in correlation with the destination.

RECEIVING NODE:
Steps:
1. Receive the packet at the receiver stream.
2. Get the current simulation time.
3. Check for packets at the other receiver streams also.
4. Increment the number of packets depending upon the number of packets that arrive.
5. Get the initial time at which the packet was sent out.
6. If initial time is –1, then the packet generated is a source quench message.
7. Get source and destination addresses and find the difference between the initialisation time and arrival time.
8. If the time difference is low for the first time, record a note about it. If this situation continues then the source quench message has to be generated.

Create a packet and assign the destination and source addresses along with the time value as –1.

9. Transmit the source quench message.
10. If the time difference is not high, the message could be positively received.

RESULTS

Analysis Scheme of the Model

The SQ message to the source controls the rate of data transmission onto the network and so directly impacts congestion control. The time taken for the packet to reach the destination from the source is calculated. If the source generates packets faster than the receiving capacity of the destination, the SQ message is sent to the source by the receiver. The source on receiving this SQ message reduces the rate of transfer of packets.

The model is simulated with various settings in the process model. The effect of data flow is analysed from one node to another with different round trip time. It has been found that the effect of changing the round trip time affects the whole model performance.

The following base models have been used in the simulation:

1. The error correction model: Point-to-Point error correction model (*dpt-ecc*).
2. The error allocation model: Point-to-Point error allocation model (*dpt-err*).
3. The propagation delay model: Whenever a point-to-point transmitter initiates a transmission, the computation of propagation delay is performed in order to determine the time required for the packet to reach the receiver in the destination node. A Point-to-Point propagation delay model is applied here.
4. The transmission delay model: This model is used to represent the simulated time interval that the transmitter requires to completely process and transmit the packet. A Point-to-Point transmission delay model is applied here.

The network was analysed for different data rates available in OPNET such as 4800bps, 9600bps and 19200bps and statistics obtained. These statistics were updated at intervals of simulated time determined by the *Update Intvl* (Update Interval) attribute set by the user in the simulation dialog box. The following statistics are presented in each update:

SQ Time: If the packet takes more than this time, then the source is informed to reduce its speed of packet transfer. The source will keep on increasing the speed of transmission at a gradual speed until it receives an SQ message.

SIM_T: The current simulation time in seconds.

REAL_T: The real running time of the simulation.

EV: The total number of events processed. An event usually signifies the arrival of a packet or an interrupt at a module, triggering some kind of response within the module.

AV_EV/S: The average number of simulation events per second of real time (measured since the beginning of the simulation).

IN_EV/S: The instantaneous number of simulation events per second

(measured using real time over the last update interval).

AV_EFF: The average efficiency of the simulation run. This is the simulation time divided by the real time.

The network under different data rates has been analysed with various values of *time_difference,* where time_difference = packet arrival time - packet generation time.

From the data presented in Table 1, 2 and 3, the followings were inferred:

1. The higher the data transmission rate, the higher the number of packets involved in congestion.
2. An increase in the data transmission rate results in a decrease in the time taken for the packet to reach the destination from the source.
3. The number of operations that ought to be performed on a packet keeps on increasing with the increase in the time duration.

TABLE 1
Packet information for 4800 bps

SQ Time	SIM_T	Real Time	EV	AV_EV/S	AV_EFF	Packets regenerated
0.7	1000	1.8	15160	8256	544.657	0
1.0	1000	1.8	15160	8198	540.83	0
1.5	1000	1.8	15160	8317	548.656	0
2.0	1000	1.8	15160	8288	546.707	0
2.00001	1000	1.9	15277	8051	527.057	9
2.001	1000	2.3	16402	7278	443.089	121
2.1	1000	2.2	16431	7411	451.089	124
2.25	1000	2.2	16487	7414	449.689	129
2.5	1000	2.3	16581	7315	441.217	141

TABLE 2
Packet information for 9600 bps

SQ Time	SIM_T	Real Time	EV	AV_EV/S	AV_EFF	Packets regenerated
0.0	1000	1.9	15175	8186	539.454	0
0.25	1000	1.8	15175	8516	561.24	0
0.5	1000	1.8	15175	8311	547.684	0
0.75	1000	1.9	15175	8111	534.528	0
1.0	1000	1.9	15175	8161	537.806	0
1.0005	1000	2.5	17225	6945	403.228	352
1.025	1000	2.4	17231	7046	408.916	355
1.05	1000	2.5	17238	6966	404.16	356
1.1	1000	2.5	17264	7031	407.313	360
1.2	1000	2.6	17322	6791	392.093	371
1.25	1000	2.5	17342	6921	399.131	373

TABLE 3
Packet information for 19200 bps

SQ Time	SIM_T	Real Time	EV	AV_EV/S	AV_EFF	Packets regenerated
0.25	1000	1.7	15178	8700	573.243	0
0.5	1000	1.8	15178	8381	552.188	0
0.7	1000	2.7	17844	6699	375.442	200
0.75	1000	2.6	17886	6782	379.195	204
1.0	1000	2.7	18153	6643	365.947	231
1.0005	1000	2.8	18153	6584	362.709	231
1.1	1000	2.7	18161	6648	366.096	233
1.2	1000	2.7	18178	6773	372.646	236

Therefore, to produce an SQ message, the time difference between the source and destination of packets was upset at different numbers of seconds for different data rate systems. The implication of the data from these tables is that when the SQ is issued and when the time taken for the data to reach the destination is less, there is a lesser likelihood of a collision. It is a tradeoff between placing the SQ time and the total time. The time impact on the network is due to two things: travel time and the processing time. The value of the SQ time is upset with regard to the set of operations or rules needed to make packet-filtering decisions. The number of rules varies according to the need of the network. If the number of computations that needs to be performed to take an action on the packet is high, then the processing time increases. This causes the time taken for the packet to reach the destination to increase. As the SQ time is based on the time required for the packet to move from source to destination, this increase in processing affects the SQ time also. In turn, causing the total time, for the packet to reach destination from source, to change based on the number of ACL rules as processing is done on-the-fly in the case of active networks. Thus with different access lists, the SQ time will also vary to cater for the changes in total time. Minimising the SQ time will safeguard the system from congestion but at the same time will affect the efficiency of network usage. On the other hand, fixing the SQ timer at a high value makes the network prone to congestion while resulting in high network utilisation. The use of source quench in an appropriate manner (in accordance with the network capability) reduces the probability of collisions and congestion.

The simulated system was tested by varying inter-arrival times (the time gap between the transmission of two consecutive packets) between packets under passive network criteria in order to identify the impact of varying packet inter-arrival time on the network performance. There was an increase in the End-to-End (ETE) delay when packets were transmitted with short inter-arrival time but network utilisation has increased because of faster packet transmission rate, as shown in *Fig. 5*.

The increase in the ETE delay indicates a possibility of congestion in the network. If the packet inter-arrival time is small, then there will be more packets

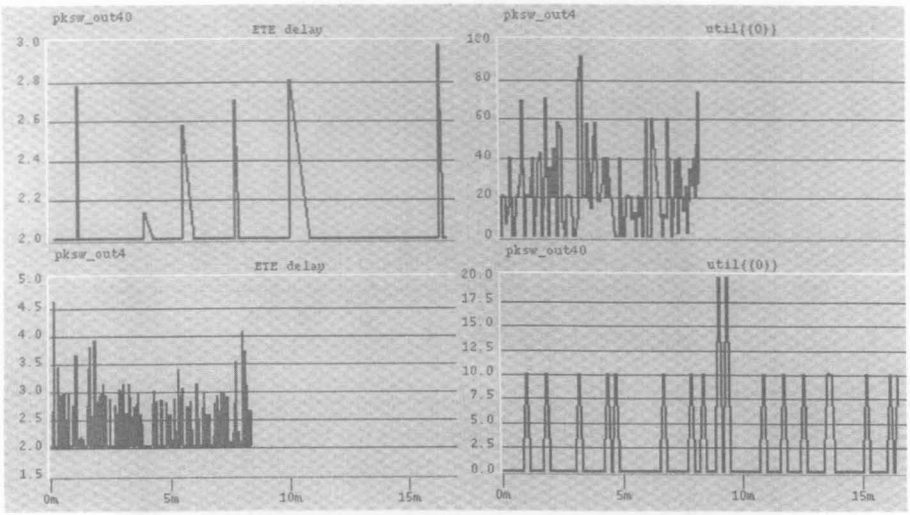


Fig. 5: ETE Delay & Utilisation with inter-arrival times as 4 and 40 sec. (top-right-graph: Inter-arrival time of 4 sec; bottom-right-graph: Inter-arrival time of 40 sec) (x-axis-left-graph: Simulation time; y-axis-left-graph: ETE delay time) (x-axis-right-graph: Simulation time; y-axis-right-graph: Utilisation)

in the network causing higher network utilisation and ETE delay density. The above graph helps us to infer that if the packets are sent with small inter-arrival time, there is a high possibility for congestion as well as high network utilisation.

Congestion is not due only to the transfer time from source to destination but also due to the IP packet filtering process at the hub. The process of identifying the situation of congestion in the network was done at the destination end as well as at the hub and the results were compared. The impact of the network congestion found at the end hosts has an equivalent impact on the hub too, as shown in *Fig. 6.* This makes it clear that if we take an action at the hub on the congestion control, it will reduce the impact of congestion throughout the network. The analysis was done when packets were generated at various packet inter-arrival times.

After identifying that the presence of some actions at the hub makes an impact on the network, we added the code to send the SQ message from the hub to source in case there occurs a very low ETE delay. Because of the presence of SQ action at the hub itself, we can see from *Fig. 7* that the ETE delay towards the node end is reduced.

A highly congested network may end up in collision and loss of packets. To avoid this situation, we handled the congestion scenario at the router level itself. Comparing the top two graphs in *Fig. 7,* we can identify that the ETE delay is reduced and the density of ETE delay is also reduced. This is due to the fact that the intermediate router takes action and sometimes destroys the packet. Therefore, it is clear that the performance of the system improves with the inclusion of Source Quench at the router level.

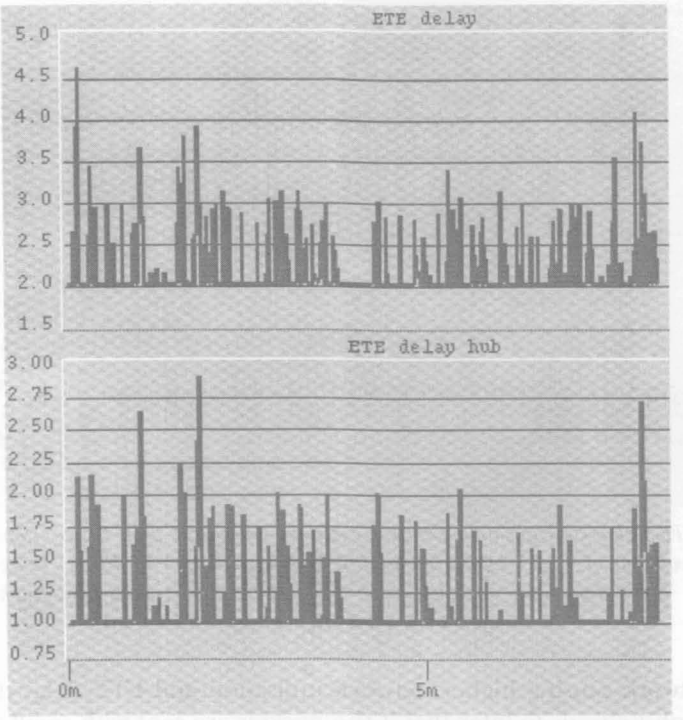


Fig. 6: ETE delay at Hub and Node (top-graph: ETE delay between source and destination; bottom-graph: ETE delay between source and hub) (x-axis: Simulation time; y-axis: ETE delay time)

DISCUSSION

Analysis of Various Congestion Control Schemes:

1. Rate Adaptation Protocol: Round trip delay (r) is measured once every round-trip-time. An increase in '*r*' is considered to be the situation of congestion and the window W is reduced to half. At the same time, if there is no increase in *r*, then the window size is increased by 1.

$$W \leftarrow W + \delta W$$
$$W \leftarrow cW;\ c = 0.5 \qquad (1)$$

If c is small or W is large at its peak, then the scheme would loose the available bandwidth except when W is at its peak for few cycles. So, the utilisation of the bandwidth is not appropriate. Here, the net idle time is based on the bottleneck server and also on the idle due to slow increase of the window size. At i^{th} period, the idle time due to slow increase of window

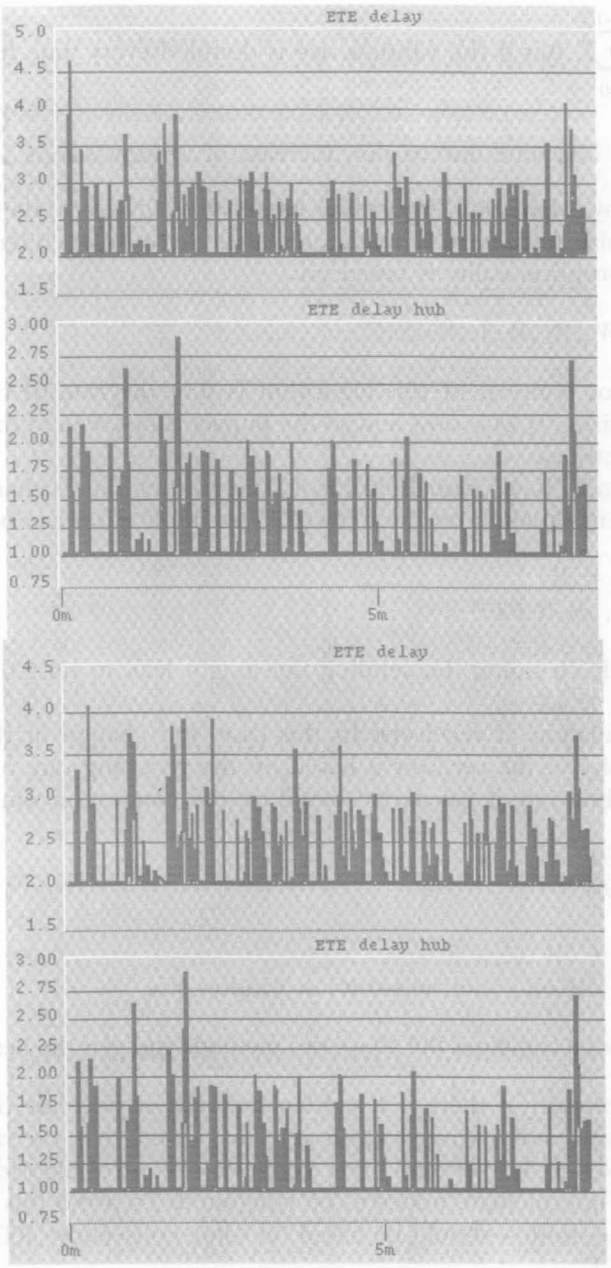


Fig. 7: ETE delay at Ñode and Hub (top-graph: without SQ at Hub; bottom-graph: with SQ at Hub)
(x-axis: simulation time; y-axis: ETE delay time)
(ETE delay: ETE delay between Source and Destination; ETE delay Hub: ETE delay between Source and Hub)

size is $\sum_{i=0}^{V-1} i$. But if the window size is doubled every time for an increase, then the idle time due to slow increase of window size is $\sum_{i=0}^{V-1}\left(V-2^i\right)$.

2. Loss-Delay Adaptation Algorithm: As per this algorithm, the change of the window size (at time t) is made only after three independent analysis and so, the proportionality is based on:

$$W_t \, \alpha [W_{t-1}, W_{t-2}, W_{t-3}] \tag{2}$$

The major concern of this algorithm is that the rate of change on the window size, $\partial W/\partial t$ is slow and so the impact is not seen at the proper time of the network.

3. TCP-Friendly Rate Control Protocol: The sending rate is changed based on the current situation on the network. This implication can be expressed as

$$\lambda \leftarrow \lambda + \partial\lambda$$
$$\text{Where, } \partial\lambda \; \alpha \; RTT \tag{3}$$

Frequently changing the sending rate might lead to unstable state of the network.

4. TCP Emulation at receivers: In this case, the change of the congestion window size at the receiver is based on the receiving rate of the receiver. The sending rate is calculated based on the congestion window size.

$$\partial W \; \alpha \; \partial\lambda_r$$
$$\partial\lambda_s \; \alpha \; \partial W$$
$$\text{So,}$$
$$\lambda \leftarrow \lambda + \delta W$$
$$\text{Where } \delta W \text{ only indicates the last window size} \tag{4}$$

5. Our proposal combines the above two methods and gets the best out of them.

Special Scenario: If there is a sudden increase or decrease of the sending rate especially when the window size is at its maximum, then the rate adaptation algorithm and the TCP-Friendly rate congestion control algorithm cause the buffer at the intermediate nodes to be congested. To avoid this situation, the change on the window should be based on some consecutive RTT and not on individual RTT.

Along with the special scenario indicated, our proposed algorithm is not impacted by idle time and slow response like the Rapid adaptation and Loss-Delay algorithms. The possibility of unstable condition (as in TCP-Friendly rate algorithm) is avoided in our proposed system. Taking into consideration the receiving rate and the state of the network helps the proposed algorithm to be more stable.

Impact on ETE Delay

Using *Fig. 5*, we plan to analyse the impact on ETE delay by varying inter-arrival times, as 4 & 40 seconds. Here, the number of levels (a) is 2 (inter-arrival time as 4 & 40 seconds), and the number of replications (r) is 8. The differences between means and the overall mean give the impact on ETE delay.

TABLE 4
Impact on ETE delay

	Out4	Out40	
Mean	2.92375	2.215	2.569375
Impact on ETE delay	0.354375	-0.354375	

We can find out the experimental errors by finding the difference between the estimated response and the measured response using the equation:

$$\text{Experimental Error } e_{ij} = y_{ij} - \bar{y}_{ij} - \mu - \alpha_j \quad (5)$$

$$\text{Sum of Squared Errors SSE} = \sum_{i=1}^{b}\sum_{j=1}^{a} e_{ij}^2$$

where yij = measured response
$\bar{y}ij$ = estimated response
μ = mean of all the measured response values
α_j = mean of the each Column j

The percentage of variation explained by varying inter-arrival times (as 4 and 40 seconds) is 39.03%. The unexplained variation is 60.96%.

Looking at these percentages, it is found that varying inter-arrival times considerably impacts the ETE delay of the system.

Impact on Utilisation of the Network

Using *Fig. 5*, we plan to analyse the impact on utilisation by varying inter-arrival times as 4 & 40 seconds. Here, the number of levels (a) is 2 (inter-arrival time as 4 & 40 seconds), and the number of replications (r) is 6. The differences between means and the overall mean give the impact on utilisation.

TABLE 5
Impact on utilisation

	Out40	Out4	
Mean	6.8543333	55	30.9271667
Impact on utilisation	-24.072834	24.072833	

Using Equation 5, we can find out the experimental errors by finding the difference between the estimated response and the measured response.

The percentage of variation explained by varying inter-arrival times is 76.55%. The unexplained variation is 23.44%.

Looking at these percentages, it is found that varying inter-arrival times considerably impacts the utilisation of the network. Thus, we can visualise that the impact of the inter-arrival time on the utilisation is more than that towards ETE delay of the system.

Impact on ETE delay at Hub and Node

Using *Fig. 6*, we plan to analyse the impact on ETE delay at the hub and the node. Here, the number of levels (a) is 2 (ETE delay at hub, ETE delay at node), and the number of replications (r) is 16. The differences between means and the overall mean give the impact on ETE delay both at hub and node.

Using Equation 5, we can find out the experimental errors by finding the difference between the estimated response and the measured response.

The percentage of variation explained by the ETE delay at the node/hub is 62.025%. The unexplained variation is 37.97%.

Looking at these percentages, it is found that there is considerable impact on the ETE delay at the hub with respect to the ETE delay at the node.

TABLE 6
Impact on ETE Delay at hub and node

	Node	Hub	
Mean	2.97625	1.7425	2.359375
Impact on ETE delay	0.616875	-0.616875	

Impact on ETE Delay at Hub and Node without SQ

Using *Fig. 7*, we plan to analyse the impact of ETE delay at the node and hub without SQ enabled. Here, the number of levels (a) is 2 (ETE delay at node, ETE delay at hub), and the number of replications (r) is 16. The differences between means and the overall mean give the impact on ETE delay at hub and node without SQ.

Using Equation 5, we can find out the experimental errors by finding the difference between the estimated response and the measured response.

The percentage of variation explained by the ETE delay at the node and hub is 56.47%. The unexplained variation is 43.53%.

Looking at these percentages, it is found that the ETE delay at the node has significant impact on the ETE delay at the hub without SQ being enabled.

TABLE 7
Impact on ETE Delay at hub and node without SQ

	ETE Delay	HUB	
Mean	2.97625	1.846875	2.4115625
Impact on ETE delay	0.5646875	-0.564688	

Impact on ETE Delay at Hub and Node with SQ

Using *Fig. 7*, we plan to analyse the impact on ETE delay at the node and hub with SQ enabled. Here, the number of levels (a) is 2 (ETE delay at node, ETE delay at hub), and the number of replications (r) is 13. The differences between means and the overall mean give the impact on ETE delay at hub and node with SQ.

Using Equation 5, we can find out the experimental errors by finding the difference between the estimated response and the measured response.

The percentage of variation explained by the ETE Delay at node/hub is 54.28%. The unexplained variation is 45.72%.

Looking at these percentages, it is found that the ETE delay at the node has significant impact on the ETE delay at the hub with SQ being enabled. Thus, we can visualise that there is significant impact on the ETE delay at the hub by the ETE delay at the node irrespective of the SQ being enabled or not.

TABLE 8
Impact on ETE Delay at hub and node with SQ

	HUB	ETE Delay	
Mean	1.9284615	3.0684615	2.49846154
Impact on ETE delay	-0.57	0.57	

Impact on ETE Delay at Hub with and without SQ

Using *Fig. 7*, we plan to analyse the impact on ETE delay at the hub with and without SQ enabled. Here, the number of levels (a) is 2 (with SQ, without SQ), and the number of replications (r) is 17. The differences between means and the overall mean give the impact on ETE delay at hub with and without SQ.

Using Equation 5, we can find out the experimental errors by finding the difference between the estimated response and the measured response.

The percentage of variation explained with or without using SQ is 0.031%. The unexplained variation is 99.97%.

Looking at these percentages, it is found that the variation in having or not having SQ does not impact on the ETE delay at the hub.

TABLE 9
Impact on ETE Delay at hub with and without SQ

	With SQ	Without SQ	
Mean	1.7811765	1.7652941	1.77323529
impact on ETE delay	0.0079412	-0.007941	

Impact on ETE Delay at Node with and without SQ

Using *Fig. 7*, we plan to analyse the impact of ETE delay at the node with and without SQ enabled. Here, the number of levels (a) is 2 (with SQ, without SQ), and the number of replications (r) is 13. The differences between means and the overall mean give the impact on ETE delay at node with and without SQ.

Using Equation 5, we can find out the experimental errors by finding the difference between the estimated response and the measured response.

The percentage of variation explained by using or non-using of SQ algorithm is 0.3076%. The unexplained variation is 99.69%.

Looking at these percentages, it is found that the variation in having or not having SQ does not impact on the ETE delay at the node.

TABLE 10
Impact on ETE Delay at node with and without SQ

	With SQ	Without SQ	
Mean	3.0353846	2.9669231	3.00115385
Impact on ETE delay	0.0342308	-0.034231	

CONCLUSION

Efficiency of a network can be improved by providing improved quality of service. Thus, enhancement to the quality of service of Transmission Control Protocol (TCP) is required to improve network performance. Enhancement of QoS using Source Quench messages to control the speed at which the source sends data to the receiver was implemented. The retransmission time for the network was set by first analysing the time taken by a packet to reach the destination from the source, along with the processing time.

From the discussion, we have indicated that the processing of the packets to check for congestion was done on intermediate nodes, which is using the processing on-the-fly concept of active networks. With the active networks oriented congestion control, our discussion states that there is significant impact on the ETE delay at the hub by the ETE delay at the node irrespective of the SQ being enabled or not. Being a simulated system, the communication time is not as appropriate as the case of a real system and also the processing is done on one processor. The possibility of congestions is identified earlier at the routers, and the necessary action is taken.

REFERENCES

GREGORY, G. FINN. 1995. A connectionless congestion control algorithm. *Proceedings of ACM SIGCOMM* **19(5):** 12-30.

FLOYD, S., J. PADHYE and J. WIDMER. 2000. Equation-based congestion control for unicast applications. *Proc. ACM SIGCOMM*, p. 43-56. Stockholm, Sweden.

LAKSHMAN, T.V., and U. MADHOW. 1997. The performance of TCP/IP for networks with high Bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking* **5(3):** 336-350.

PARTRIDGE, C. and T. J. SHEPARD. 1997. TCP/IP performance over satellite links. *IEEE Network* **11(5):** 44-49.

REJAIE, R., M. HANDLEY and D. ESTRIN. 1999. Rap: An end-to-end rate-based congestion control mechanism for real-time streams in the internet. *Proceedings of IEEE INFOCOM* p. 1337-1345.

RHEE, I., V. OZDEMIR and Y. YI. 2000. TEAR: TCP emulation at receivers – flow control for multimedia streaming. *Tech. Report*, Department of Computer Science, NCSU.

SISALEM, D. and A. WOLISZ. 2000. LDA+ TCP-friendly adaptation: A measurement and comparison study. *IEEE International Conference on Multimedia and Expo* (III), p. 1619-1622.

TANENBAUM, A. S. 1996. *Computer Networks*. New Jersey: Prentice-Hall International.

WANG, Z. and J. CROWCOFT. 1992. Eliminating periodic packet losses in the 4.3-Tahoe BSD TCP congestion control algorithm. *Computer Communication Review* **22(2):** 9-13.

WIDMER, J., R. DENDA and M. MNAUVE. 2001. A survey on TCP-friendly congestion control. *IEEE Network*, May/June: 28-37.