Traffic Monitoring using Isolate and Control Algorithm in TCP/AQM Networks

Nithyashree G D[1] Shashidhara M S[2]
[1,2]Department of Computer Science & Engineering
[1,2]CIT, Gubbi, Tumkur

Abstract— To support the myriad of envisioned communication products of the future, there is a need to develop a network infrastructure that can provide larger bandwidth, with better control of quality of service (QoS). However, with increasing demand for applications running over packet networks, congestion at the intermediate nodes (e.g., routers and switches) can be a serious problem. Consequences include long delays, large delay variation and high packet loss rates. Different solutions requiring varying levels of modification to the currently used algorithms have been proposed both for responsive (e.g., TCP) and unresponsive (e.g., UDP) protocols. However, most of the solutions are either too complicated to implement in real life or not general enough to be applicable to an arbitrary network topology. In this project a new AQM algorithm ICU AQM (Isolate and Control- AQM) which effectively detects unresponsive flows and controls them based on the network load status is proposed in order to strengthen the robustness of Internet against unresponsive flows. ICU AQM algorithm is used to detect and isolate the unresponsive flows which in turn results in protection to responsive flows. The project work is simulated using NS2. The results are compared with other AQM algorithms to show the increase in the performance of the proposed algorithm.

Key words: Bandwidth, Isolate and Control- AQM, Network Simulator 2 (NS2), Quality of service (QoS), Transmission Control Protocol (TCP), User Datagram Protocol (UDP)

I. INTRODUCTION

The Internet, traditionally FTP, e-mail and Web traffic, is increasingly supporting emerging applications such as IP telephony, video conferencing and online games. These new genres of applications have different requirements in terms of throughput and delay than traditional applications. For example, interactive multimedia applications, unlike traditional applications, have more stringent delay constraints and less stringent loss constraints. Moreover, with the use of repair techniques packet losses can be partially or fully concealed, enabling multimedia applications to operate over a wide range of losses, and leaving end-to-end delays as the major impediment to acceptable quality. Thus, interactive multimedia applications prefer smaller queues in Internet routers. Web traffic is moderately sensitive to delay as well as throughput and hence, it falls in the middle of the spectrum and prefers medium queues in Internet routers.

When the number of packets dumped into the network is within the carrying capacity, they all are delivered, expect a few that have to be rejected due to transmission errors. And then the number delivered is proportional to the number of packets sent. However, as traffic increases too far, the routers are no longer able to cope, and they begin to lose packets. At very high traffic, performance collapse completely, and almost no packet is delivered. Unfortunately, current Internet routers are not able to provide a choice in Quality of Service (QoS). Most of the current Active Queue Management (AQM) techniques focus on providing higher throughput at the router without much consideration for queuing delays.

The goal of good Queue Management schemes is to allocate bandwidth fairly among different traffic sources that may have different service requirements, while maximizing service utilization. Therefore, it is necessary for a router to maintain a small queue length with enough buffer capacity to absorb the bursty data traffic. In addition, it is necessary to detect congestion before it becomes a problem (i.e., before overflows) in order to control congestion efficiently and keep the network stable. One possible solution to overcome the drawbacks of the drop tail scheme is to drop packets before a queue becomes full so that a source can respond to congestion before buffers overflow. This approach is called Active Queue Management (AQM) (3).

II. EXISTING SYSTEM

AQM algorithms for Unresponsive Flows During periods of congestion, responsive flows reduce the load they generate while unresponsive flows may or may not do so. As a result, responsive flows can suffer from starvation effect while unresponsive flows benefit from their greedy nature. Many of the proposed approaches to this problem focus on encouraging end systems to use responsive protocols by deploying mechanisms in network switches that penalize unresponsive traffic. This chapter gives the overview of the many algorithms for Active Queue Management for managing the unresponsive flows on the router. The sections discuss briefly about the working of RED, REM, CLUE and other unresponsive flow related algorithms and also a brief working of ICU-AQM algorithm.

Random Early Detection (RED)The AQM algorithm introduced by Jacobson and Floyd, Random Early Detection (RED), is the keystone of current studies. Although original RED was believed and had been shown in recent publications to have serious performance problems, it has been recommended by the Internet Engineering Task Force (IETF) as the default active queue management scheme for next generation networks with the following statement. "Internet routers should implement some active queue management mechanism to manage queue lengths, reduce end-to-end latency, reduce packet dropping, and avoid lock-out phenomena within the Internet. The default mechanism for managing queue lengths to meet these goals is RED. Unless a developer has reasons to provide another equivalent mechanism, it is recommended RED be used." Random Early Detection (RED) [2], in contrast to the traditional queue management disciplines such as Drop-Tail

(which drops the packets only when the buffer is full), drops the packets before the buffer overflows. Although it is designed to work with a single queue, it can easily be made to work with multiple queue systems like fair queuing such that RED is applied to each queue independently.

Random Early Detection has two crucial objectives. First of all, as it can be understood from the words "Early" and "Detection", it tries to keep the queue away from the congestion limit by using some signals as the indicators of congestion. Secondly, it drops the packets in a random manner according to a probability function which increases as the average queue size increases. This ensures that RED is not biased against any connection or any bursty traffic and prevents the monopolization of some flows. Moreover, since the packets are dropped randomly according to the average queue size rather than the instantaneous queue size, a better notion of congestion is assured; if the average queue size remained recently high, there is a high probability of persistent congestion. Also RED eliminates the global synchronization and the high latency problems that traditional queues faced by controlling the average queue length.

Random Early Marking (REM)A new Rate based flow control approach called REM has been proposed by V. Lapsley and D. Low [8]. These techniques require only minimal changes to existing TCP host behavior, and RED active queue management routers. Thus, this collection of techniques is called as Random Early Marking (REM). Here, these techniques enable implementing flow control in an Explicit Congestion Notification capable TCP/IP network.

Random Early Marking (REM) consists of a link algorithm, that probabilistically marks packets inside the network, and a source algorithm, that adapts source rate to observed marking. Marking allows a source to estimate its path congestion measure and adjusts its rate in a way that aligns individual optimality with social optimality. The marking probability is exponential in a link congestion measure, so that the end-to-end marking probability is exponential in a path congestion measure. Because of the finer measure of congestion provided by REM, sources do not constantly probe the network for spare capacity, but settle around a globally optimal equilibrium, thus avoiding the perpetual cycle of sinking into and recovering from congestion.

The first idea of REM is to stabilize both the input rate around link capacity and the queue around a small target, regardless of the number of users sharing the link. Each output queue that implements REM maintains a variable we call 'price' as a congestion measure. This variable is used to determine the marking probability, as explained in the next subsection. Price is updated, periodically or asynchronously, based on rate mismatch (i.e., difference between input rate and link capacity) and queue mismatch (i.e., difference between queue length and target).

The second idea of REM is to use the sum of the link prices along a path as a measure of congestion in the path, and to embed it into the end-to-end marking probability that can be observed at the source. The output queue marks each arrival packet that is not already marked at an upstream queue, with a probability that is exponentially increasing in the current price. The exponential form of the marking probability is critical in a large network where the end-to-end marking probability for a packet that traverses multiple congested links from source to destination depends on the link marking probability at every link in the path. When, individual link marking probability is exponential in its link price, this end-to-end marking probability will be exponentially increasing in the sum of the link prices at all the congested links in its path.

Stochastic RED (StoRED)Shan Chen, Zhen Zhou and Brahim Bensaou proposes a novel fair queue management algorithm called Stochastic RED (StoRED), inspired by the well known stochastic fair queuing and based on the Random Early Detection scheme which controls both responsive and unresponsive flows [16]. StoRED applies the idea of time varying hashing. Only misbehaving flows get significantly disciplined by Stochastic RED.

It is called "stochastic" because it does not really distinguish the flows accurately. In StoRED, the arriving traffic is divided by the router into a limited number of "counting bins" using a hashing algorithm. On the arrival of each packet at the queue, a hash function is used to assign the packet to one of the bins based on the flow information. By using an appropriate hash function, StoRED dispatches the packets of the different flows to the set of bins. With a given hash function, the packets of the same flow are mapped to the same bin. Therefore, when the flow is unresponsive, the bin load increases dramatically. StoRED estimates the bin loads and uses these loads to penalize flows that map to each bin according to the load of the associated bin. Thus unresponsive flows experience a larger packet drop probability. However, because of the hashing and the limited number of bins, multiple flows may end up associated with the same bin. Thus flows that share a bin with an unresponsive flow are punished unnecessarily. To prevent this situation from becoming noticeable, StoRED changes its hashing function often enough so that the time span for any two flows to collide into the same bin is within several seconds. In the long run, only misbehaving flows get significantly disciplined by StoRED.

Stochastic Fair Blue (SFB)Wu-chang Feng, Dilip D. Kandlur, Debanjan Saha and Kang G. Shin have proposed Stochastic Fair Blue (SFB) which effectively handles unresponsive flows using an extremely small amount of state information [17]. SFB is a novel technique for enforcing fairness among a large number of flows. SFB is a simple modification of the BLUE algorithm. BLUE is a fundamentally different queue management algorithm which uses a single marking probability to manage congestion.

SFB detects and rate-limits non-responsive flows through the use of a marking probability derived from the BLUE queue management algorithm and a Bloom filter. Bloom filters are commonly used in word processing software applications as an efficient means to do spell-checking. They are also used in web caches to efficiently determine the existence of an object in the cache. The idea behind Bloom filters is to use L levels of bins with each level containing N bins. For each level, an independent hash function is used to hash a particular object (URL string, English word, TCP/IP connection ID, etc.) into one of the N bins. Each object is then classified and identified by the bins it maps into in each level. Since an object can map into N

possible values at each level, an object is identified by an L-tuple of numbers which range from 1 to N.

SFB is shown to effectively handle unresponsive flows using an extremely small amount of state information. But, misclassification of flows limits the ability of SFB to protect well behaved TCP flows.

III. PROPOSED SYSTEM

ICU (Isolate and Control)ICU AQM (Isolate and Control-AQM) effectively detects unresponsive flows and controls them based on the network load status. Here a flow isolator isolates the unresponsive flows from a traffic mix of responsive flows and unresponsive flows, using the traffic arrival rate and loss ratio of the incoming flows. Network load status is checked and if there is a load unbalance, the adaptive packet dropping policy is triggered automatically wherein the unresponsive flows are shifted to a drop window. For every small interval, depending on the status of network load, the size of the drop window is either incremented or decremented. If the size of the window increases beyond a threshold value, the packets will be dropped.

The algorithm consists of a flow isolator isolates the unresponsive flows from a traffic mix of responsive and unresponsive flows using the traffic arrival rate and loss ratio of the incoming flows. Flow Arrival Rate estimator which calculates $EAR_i(t)$ of the incoming flows at the edge router and fixes this rate into packet label. Unresponsive flow detector checks for the load un-balance in the network calls the adaptive packet dropping policy module.Adaptive packet dropping policy limits the sending rate if the flow is unresponsive.

The design of an active queue management system for unresponsive flows based on adaptive dropping policies is given in the ICU algorithm. In this active queue management system, a flow isolator isolates the unresponsive flows from a traffic mix of responsive and unresponsive flows using the traffic arrival rate and loss ratio of the incoming flows. Then it checks for load un-balance of the network and automatically triggers the adaptive packet dropping policy. In our architecture, the rates EARi(t) of the incoming flows are calculated approximately at the edge routers and then these rates are fixed into the packet labels. At every edge router, we use exponential averaging to estimate the rate of a flow. Back to-back probe packets are sent for a small sample interval of TS seconds to infer a link loss by computing the correlation of a packet loss within a set of probe packets at different destinations.

Source sends a series of probe packets along a path R_i to the destination, with no delay among the transmissions of successive packets. The triggering policy raises the Adaptive Dropping Policy when the system is unbalanced. If the current queue length Q_{len} of node is more than the threshold Q_{th}, then the system is considered as unbalance and the load balancer is invoked(1).

Figure 3.1 shows the schematic of ICU algorithm work. In this active queue management system, a flow isolator isolates the unresponsive flows from a traffic mix of responsive and unresponsive flows using the traffic arrival rate and loss ratio of the incoming flows. The flows will be both TCP and UDP based.

The system analyzer will estimate the flow property and estimates the balancing condition for that flow based on the packet. Then it checks for load un-balance of the flow traffic and if the flow is unbalanced they the protocol automatically triggers the adaptive packet dropping policy (ADP) to identify the packet to be dropped and records the drop probability of that flow. If the packet is dropped then the drop probability is decreased otherwise the drop probability is increased.

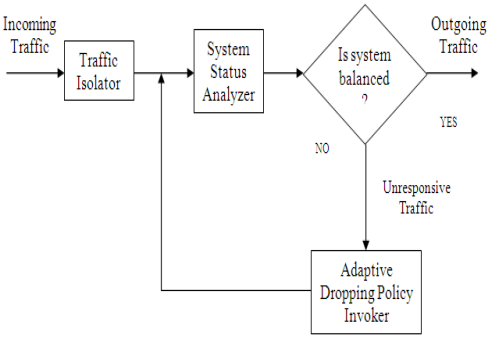

Fig. 3.1: A Schematic Diagram of ADP active queue Management System

IV. IMPLEMENTATION

Active Queue Management (AQM) mechanisms are link algorithms and are deployed inside the network, i.e. in the routers, to regulate the flows. By sending congestion signals in a proactive manner, an AQM technique makes attempt to prevent congestion and control the queue length. The primary goal of this project is to achieve the better performance in QoS of responsive flows over the unresponsive flows.

Here a flow isolator isolates the unresponsive flows from a traffic mix of responsive flows and unresponsive flows, using the traffic arrival rate and loss ratio of the incoming flows.

The design of ICU includes adaptive dropping policies. The flow isolator module isolates the responsive flows from a traffic mix of responsive and unresponsive flows using traffic arrival and loss ratio of the incoming flows. The Balancer module will check the load un-balance condition over the flows in the network and automatically calls the packet dropping policy module. The major design issues are: Performance degradation in current TCP Congestion Control in terms of multiple packet loss, low link utilization and congestion collapse. The role of the router becomes important to control congestion effectively in networks.

A. ICU Architecture

Figure 4.1 shows the schematic of ICU algorithm work. In this active queue management system, a flow isolator isolates the unresponsive flows from a traffic mix of responsive and unresponsive flows using the traffic arrival rate and loss ratio of the incoming flows. Then it checks for load un-balance of the network and automatically triggers the adaptive packet dropping policy.

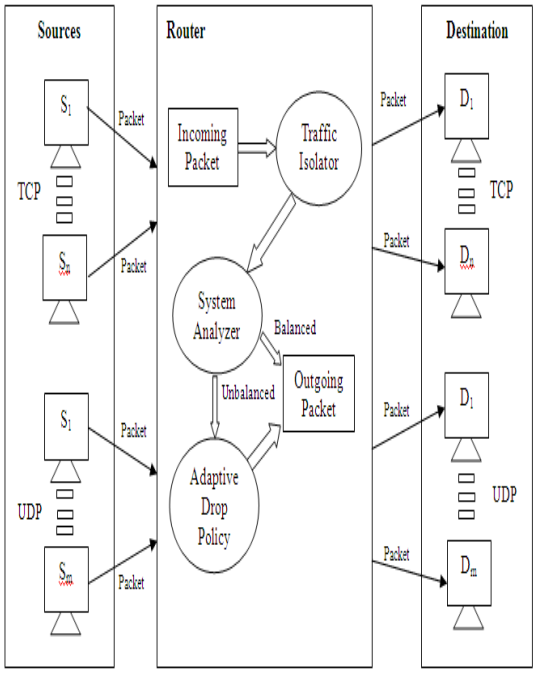

Fig. 4.1: ICU Architecture

B. System Topology

An active queue management system for unresponsive flows based on adaptive dropping policies is one of the design factors. The proposed design of active queue management system contains a flow isolator which isolates the unresponsive flows from a traffic mix of responsive and unresponsive flows using the traffic arrival rate and loss ratio of the incoming flows. Next design after isolation of flows is to check the load un-balance of the network and automatically trigger the adaptive packet dropping policy.
Figure 4.2 shows the simulation topology used , which consists of 20 nodes, with five nodes as routers. The system uses a combination TCP and UDP connections. Here the TCP connections are assumed as responsive flows and UDP connections are assumed as unresponsive flows.

The router 0 and router 1 have duplex connection with the AQM based on ICU algorithm bandwidth of 2Mbps and 25ms link delay and the rest of the connections are duplex with 5Mbps bandwidth and 25ms link delay and with AQM based on DROPTAIL algorithm. In this configuration, the link between two AQM routers (router 0 and router 1) is the identified as bottleneck link.

For each execution of the simulation provides the calculation for performance metrics such as average queue length, throughput and packet loss probability, which are used to study the mechanisms of AQM on the topology. The study for performance of ICU Active Queue Management technique is illustrated with the widespread simulations using ns-2 network simulator. According to the unresponsive flow feature of the high rate of sending data, these bursty unresponsive flows have a high probability of incurring bursts of packet loss. Therefore the difference between loss ratio due to Probe packets and loss ratio due to actual flow as one factor and arrival rate as another factor for classification.

V. RESULTS AND ANALYSIS

The Experiment is conducted using NS-2 simulator. Ns-2 stands for Network Simulator version 2. It works at packet level. Provide substantial support to simulate bunch of protocols like TCP, UDP, FTP, and HTTP [20]. NS-2 makes use of two languages TCL script for front end code and C++ for back end. Simulator works upon link front end and back end.

Figure 5.1provides the graph view for the output of the ICU along with RED, REM and CLUE algorithms, for the average link bandwidth utilization over the number of UDP flows. From the graph it is observed that the bottle neck bandwidth utilization is higher in ICU algorithm compared to other shown algorithms. It also provides the graph for the trace output of the ICU along with RED, REM and CLUE algorithms, for the average link bandwidth utilization over the number of UDP flows. From the graph it is observed that the bottle neck bandwidth utilization is higher in ICU algorithm compared to other shown algorithms. The trace file is generated for the range from 1 to 10 UDP connections and with a constant 1 TCP connection along with 10Mbps bandwidth and 5ms link delay. The CBR application running on the UDP transmits the packets at a constant rate of 1Mbps with 1000 bytes packet. The UDP source connection is 10Mbps with 5ms link delay. The queue gets build up to the full state dropping the more UDP packets and passing the TCP packets.

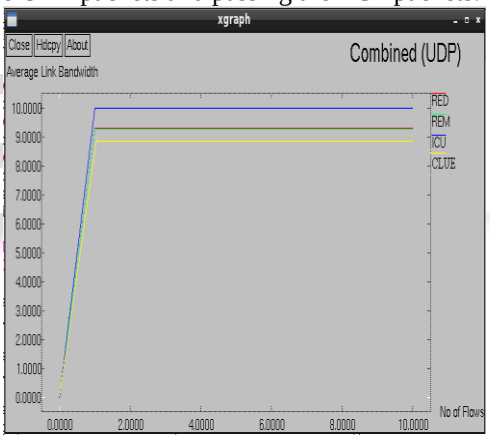

Fig. 5.1: Average Link Bandwidth Over Number of UDP Flows

The TCP source connection is 10Mbps with 5ms link delay. The queue gets build up to the full state dropping the more UDP packets and passing the TCP packets. The average link bandwidth reaches the peak value with the increase with the increase number of TCP connections. From the graph it is observed that the bottle neck bandwidth utilization is reaching to a peak of 5.33 Mbps in ICU algorithm compared to RED with 4.3 Mbps, REM with 4.57 Mbps and Clue with 5.12Mbps.

Figure 5.2 shows the graph for the average throughput over the number of TCP flows of ICU along with RED, REM and CLUE algorithms. It is observed from the graph that the average throughput of the ICU is more than the other algorithms. The trace file is generated for the range from 1 to 10 TCP connections and with a constant 1 UDP connection along with 10Mbps bandwidth and 5ms link delay. The TCP source connection is 10Mbps with 5ms link delay. The queue gets build up to the full state dropping the more UDP packets and passing the TCP packets. The

average link bandwidth reaches the peak value with the increase with the increase number of UDP connections. From the graph it is observed that the average throughput is reaching to a peak of 1220.8 Mbps in ICU algorithm compared to RED with 997.07 Mbps, REM with 1130 Mbps and Clue with 1099.8 Mbps.

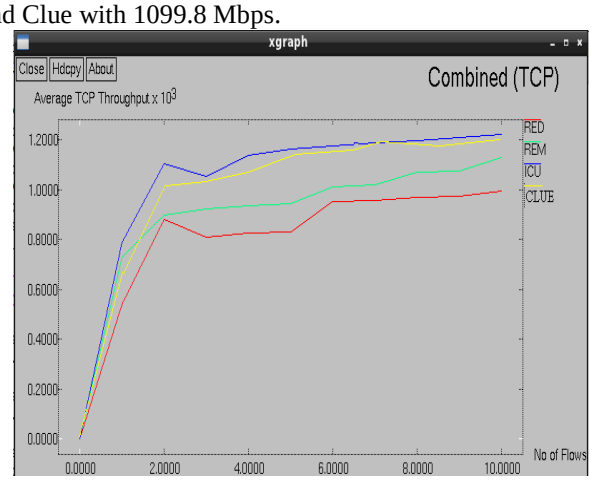

Fig. 5.2: Average Throughput over Number of TCP Flows

VI. CONCLUSION

Unresponsive flows disturb the balance of the Internet.Furthermore, the extreme unresponsive flows steal's the network resources. The new AQM algorithm- ICU is defined with the design of the algorithm which shows the problems of unresponsive flows from the point of network resource allocation. The ICU algorithm incorporated at the router classifies the traffic into responsive and unresponsive based on incoming arrival rate, which reduces the overhead due to classification based on packet header information. As ICU maintains only the state of the detected unresponsive flows, the router computation load cannot be increased.

Finally, ICU algorithm performs detection and punishment of the unresponsive flows their by it protects the responsive flows. ICU-AQM algorithms has been compared with the other algorithms like RED, REM and CLUE and analyzed with the performance parameters bandwidth and throughput for justifying the quality of service.

VII. FUTURE ENHANCEMENT

When various types of traffic flows exists in the network, this ICU algorithm can be used to protect the short-lived flows along long-lived bursty flows. The algorithm will not differentiate between the TCP and UDP flows which can be enhanced. In multimedia applications different types of data transfers/flows exist and the ICU algorithm can be extended so that it will prioritize the data's and protect the higher priority data from lower priority data transfers. The algorithm does not support for the multistream architecture thus concept can be further proceeded to handle the multistream architectures. Handling the malicious flows such as Denial of Service attacks, thus algorithm can be extended to detect and punish the malicious flows.

REFERENCE

[1] Thiruchelvi G., Raja J., "Active QueuManagement Based Adaptive Flow Control Mechanism for Unresponsive Flows", European Journal of Scientific Research ISSN 1450-216X Vol.70 No.1 (2012), pp. 67-80 EuroJournals Publishing, Inc. 2012.

[2] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance", IEEE/ACM Transactions on Networking, vol.1, no.4,pp. 397, August, 1993.

[3] G.Thiruchelvi and J.Raja, "An Adaptive Congestion Control Mechanism for Unresponsive Flows", Internet Multimedia Services Architecture and Application (IMASS), in proceedings of 2009 IEEE international conference, DEC 1-9, 2009,pp1-5.

[4] W. Feng, D. Kandlur, D. Saha, and K. management schemes," April 1999, Technical Report, CSE-TR-387-99, U. Michigan.

[5] D. Lin and R. Morris, "Dynamics of random early detection", Proceedings of the ACM SIGCOMM '97 conference on Applications, technologies, architectures, and protocols for computer communication, pp. 127-137, September 14-18, 1997, Cannes, France.

[6] Srisankar S. Kunniyur and R. Srikant, "An Adaptive Virtual Queue (AVQ) Algorithm for Active Queue Management", IEEE Transactions on Networking, vol. 12, no. 2, pp: 286 – 299, April 2004.

[7] R. Pan, B. Prabhakar, and K. Psounis., "Choke, a stateless active queue management scheme for approximating fair bandwidth allocation", In Proceedings of IEEE INFOCOM'00, pp. 942-951, Tel Aviv, Israel, March 2000.

[8] V Lapsley D., Low S., "Random early marking for Internet congestion control", in proceedings of Conference on Global telecommunications, vol.3, pp 1747- 1752, 1999.

[9] Cui-Qing Yang and Reddy, A.V.S, "A taxonomy for congestion control algorithms in packet switching networks",IEEE/Network,vol.9, no.4, pp 34-45, July/August 1995.

[10] Andrew.S.Tanenbaum, "Computer Networks- 3rd edition", Pearson education, South Africa, 2006.

[11] C.V. Hollot, Yong Liu, Vishal Misra and Don Towsley, "Unresponsive Flows and AQM Performance", Proceedings of IEEE INFOCOM, 2003.

[12] C. Padhye, K. Christensen, and W. Moreno. A New Adaptive FEC Loss Control Algorithm for Voice Over IP Applications. In Proceedings of IEEE International Performance, Computing and Communication Conference, February 2000.

[13] Yubing Wang, Mark Claypool, and Zheng Zuo. An Empirical Study of RealVideo Performance Across the Internet. In Proceedings of the ACM SIGCOMM Internet Measurement Workshop, November 2001.

[14] Alberto Leon-Garcia and Indra Widjaja, "Communication Networks- 2nd edition", published by Tata McGraw-Hill publishing Company Limited, New Delhi,2004.

[15] Shan Chen, Zhen Zhou and Brahim Bensaou, "Stochastic RED and Its Applications", in proceedings Of IEEE International Conf. on Communications, pp: 6362- 6367, 28 June, 2007.

