FLOW BASED ALGORITHM FOR DYNAMIC BUFFER ALLOCATION

SANKAR P.[1], MANU NATARAJAN[2], CHELLAMUTHU C.[3]

Virtual Clock algorithm (VC) is a traffic control algorithm that monitors the average transmission rate of packet data flow. The features of VC include guaranteed throughput, low queuing delay and firewall protection among flows. VC does not provide fairness for dynamic buffer allocation when different categories of flows are to be scheduled. Hence modification is necessary in VC to provide fair share of allocation for various flows. In this paper a modification to VC has been carried out, with a fair share of bandwidth allocation for three applications such as Video, CBR and Audio. The simulation was carried out in C++based software platform and the performance parameters like delay, throughput and packet loss ratio are calculated and analyzed.

Keywords: Computer network, Internet, Network topology, Scheduling algorithm, Video codec

1. Introduction

The increase in space requirement for network on chip design as compared to the bus based architecture requires different routing algorithms and arbitration strategies. Many researchers are working on several strategies to find the suitability of algorithms [1].The three basic blocks in a network on chip consists of router, links and a network adapter. The router basically receives packets from various links according to the packet format, based on which the packet is forwarded to other links. In addition to the physical connection the router contains logical block that implements control policies. These policies ensure deadlock free routing. The flow control involves centralized and distributed approach with no traffic congestion. The possible approach for its implementation is the use of Time Division Multiplexing (TDM) mechanism where each packet is associated with a time frame. Apart from these the escalation of applications such as voice, audio and other

[1] Jerusalem College of Engineering, Anna University, India, e-mail: sankarp1@ieee.org
[2] Meenakshi Sunderrajan College of Engineering, Anna university, India, e-mail: manu.natarajan@gmail.com
[3] RMK Engineering College, Anna University, e-mail: ccm.eee@rmkec.ac.in

services increases the use of UDP as a transport protocol in the internet. These increase of applications lead to traffic congestion in the network [2] [3]. The TCP applications have congestion control mechanisms where as UDP does not have such mechanism. The rise of UDP applications is a major concern to fairness and steadiness [4] [5]. The UDP being insensitive, do not back off in response to packet loss. This misbehavior gives rise to congestion. The queue management at the routers in the network is an ideal solution. The Drop Tail is the queue management algorithm normally used in routers [6]. This algorithm randomly distributes the losses among the flows. Random Early Detection (RED) and Flow Random Early Drop (FRED) prevent the loss of packets and also overcome the drawback of Drop Tail [7][8]. But the buffer allocation in a fairer way is not carried out by any one of these algorithms. The scheduling algorithms perform the function of packet scheduling and queue management.

Packet scheduling is the decision process used to choose which packets need to be served or dropped. The queue management refers to the disciplines that helps in regulating the occupancy of a particular queue. Packet scheduling is one of the methods for guaranteed performance of the network. It provides performance guarantee in terms of throughput, packet loss ratio, and delay. Hence, Quality of Service (QoS) guarantee can be obtained by scheduling algorithms which does queue management for both network on chip and Internet applications effectively.

The Virtual Clock (VC) is a scheduling algorithm that controls the average transmission rate of data flows and enforces users' average resource usage according to its reservation. It also provides firewall protection among individual flows and supports priority services. It was evolved from the TDM principles. Lixia Zhang describes the VC algorithm for performing two main functionalities such as data forwarding and flow monitoring. The flow of control packets based on average transmission rate is shown in Fig 1 as flow diagram [9]. The authors describe about the utility of virtual clock as a queue management algorithm and proved its efficacy in ensuring fairness to the TCP in the presence of UDP retaining the advantages of VC algorithm [10]. In this paper a flow based modification to VC has been considered to see its effectiveness when different types of flows are passed through the network and how the buffer is allocated and shared among the flows. The remnant of the paper is organized as follows. Section 2 gives an account of the related works. Section 3 gives the flow based modification of VC algorithm. Simulation, for various scenarios along with results is described in Section 4. Section 5 gives the fairness index. Finally, the conclusion is given in Section 6.

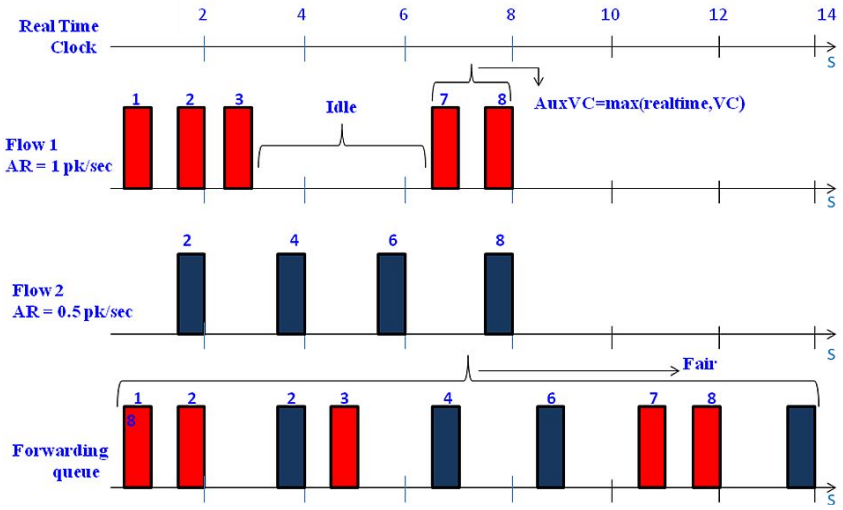

Fig. 1 Flow Diagram of Virtual Clock

2. Related works

There is an extensive literature on scheduling and queue management algorithms used in the Internet, especially in high speed networks. Abhimanyu Das [11], explains that in Internet services, when packets from two flows such as the responsive TCP and non-responsive UDP are transmitted, the responsive flows are not given proper share of resources as compared to the likes of non-responsive ones at the time of congestion. Eventually unfairness prevails among the flows which are overcome in the paper. Arnaud Legout [12], presents an analytical model to study the potential impact of video streaming strategies on the aggregate traffic and make the recommendations accordingly.

Dimitrious Stiliadis [13] describes a general model used to analyse various scheduling algorithms. Using this technique upper bounds and buffer requirements of individual sessions in various scheduling algorithms can be obtained. Geoffrey G. Xie [14] explains about the firewall protection offered by the VC algorithm along with procedures to obtain delay bounds in flows. George Varghese [15], details that the routers schedule packets on the output links in order to guarantee fairness and latency bounds. Here, VC algorithm is used to provide low end-to-end delay and guaranteed throughput.

Jianmei Chen [16], describes the principles of priority buffering incorporated into the VC algorithm. This idea mentioned in the paper provides complete isolation of different traffic classes in a sharing environment. Furthermore the paper explains that the different QoS requirements are satisfied at the same time. Nazy Alborz [17] describes the performance and efficiency of the

VC algorithm by simulating it in OPNET simulation tool. The results obtained in the paper show the wide spread areas in which VC can be applied.

3. The Flow Based Modified Virtual Clock (FMVC)

FMVC is a queue management algorithm evolved from the VC algorithm. The VC does not provide fairness among TCP and UDP flows. UDP is a non-responsive traffic protocol while TCP is responsive. During congestion TCP stops transmission while UDP continues to transmit not allowing TCP to continue its flow. Hence in VC, TCP does not get a fair share of resources. To overcome this disadvantage the FMVC is devised that introduces reservation for TCP. The network resources are fairly considered for allocation among TCP and UDP for various applications like audio, video and CBR. The buffer inside the router is spliced into various spaces and the threshold values of 25%, 50% and 25% respectively are set for the flows. If utilization by any flow exceeds the threshold value that particular flow is dropped and packets from other flows are allowed.

Working of FMVC

The modified version of VC consists of three stages. The first stage consists of classifying TCP and UDP packets for packet reservation. The second stage consists of data forwarding which is used to enqueue the packets into the outgoing queue. Third stage consists of flow monitoring for controlling each flow.

In the first stage of FMVC the packets are classified as TCP and UDP. The headers of the packets are cracked and parameters like source and destination addresses, source and destination port numbers and header sizes are obtained. These parameters help in classifying the packets as TCP and UDP.

For the first packet that is received from $flow_i$, VC_i and $auxVC_i$ are set to real time. The two parameters VC_i and $auxVC_i$ are control variables to monitor and control the flow. $Vtick_i$ is the value that is computed as the inverse of average transmission rate of $flow_i$. For every consecutive packet VC_i is advanced with $Vtick_i$ value and packets are time stamped. The time stamped packets are sent to the outgoing queue. The stamped packets are served in the increasing order. Average interval rate is the maximum number of packets a flow can transmit. It can be computed as $AIR_i = AR_i * AI_i$ for each $flow_i$, where AR_i is the average transmission rate and AI_i is the average interval of i^{th} flow which describe the data transmission behaviour. On receiving each packet the specified conditions are checked first. When the difference between VC and real time is greater than the threshold value then a warning message is sent to flow source. The warning message is a feedback information mechanism to the data flow source to adjust its transmission rate or flow parameters. Second, if VC is less than real time then VC value is enqueued to real time. This process is same as that of VC algorithm.

Algorithm of FMVC

Steps involved in FMVC are:

Step1: Common and priority header values are cracked for classification.

Step2: Queue usage is calculated.

If → Video (p)
If → usage<0.5, enqueue (p)
Else → drop (p)
If → Audio (p)
If → usage<0.25, enqueue (p)
Else → drop (p)
If → Cbr (p)
If → usage<0.25, enqueue (p)
Else → drop (p)

Step3: Packet is time stamped with AuxVC and enqueued into the queue in ascending value of the time stamp value.
If → queue length exceeds queue limit.

Step4: Drop the packet at the end of the queue.

Here 'usage' is a parameter which indicates the percentage of buffer allocated for the flow, the enqueue (p) and drop (p) are functions which will queue and drop the packets respectively. The pseudocode of FMVC given describes the manner in which it functions. The pseudocode is explained as follows:-

The pseudo-code of FMVC

variables required:
vsb : video_steal_bit
stv : stolen video size
vt : video tolerance
asb : audio_steal_bit
sta : stolen_audio_size
at : audio tolerance
db=data buffer length //stored length of the buffer
ddb= default capacity
dc=data buffer capacity //total size of the buffer.
ab= audio buffer length //stored-buffer length
ac= audio buffer capacity //total size of the buffer
vb= video buffer length //stored buffer length
vc= video buffer capacity //total size of the buffer
free=dc-db;

requirements:
1) A router is present with three type of values.
1.1) audio allocated 25%
1.2) Video allocated 50%;
1.3) DATA allocated 25%;
2) packets keeps coming in..
the pseudo-code is :
initializations :
db=0
ab=0;
vb=0;
while(1) //runs forever
{
listen for action;
if(action==input)
goto input_handling();
else
if(action==output)
goto output_handling();
}
function input_handling()
{
IF the packets=data
{
free=dc-db
if(packet.length>free)
req=packet.length-free; //the amount of extra space required
else
req=0;
if req==0 //no extra storage space required
db=db+packets.length;
else //we need to steal space from audio or video
{
if (ac-ab) > (vc-vb) //checking wether audio or video space is more
{ if (ac-ab > req and req+sta <= at) //checking if audio has enough space and if the stolen space hasnt gone above tolerence value
dc=dc+req; //adding data capacity
store packets.
db=db+packets.length
ac=ac-req; //decreasing buffer capacity of audio
asb=1;

```
} sta+=req;
else
if(vc-vb > req and req+stv<=vt)
{ dc=dc+req
vc=vc-req;
db=db+packets.length
vsb=1;
stv+=req;
}
else //stealing from both audio and video.
if((ac-ab)+(vc-vb)>=req req+sta+stv <=at+vt)
{
dc=dc+ req-(at-sta)
ac=ac-(req-(at-sta))
asb=1;
stb=stb+(req-(at-sta))
req=req-(sta+at)
dc=dc+req;
vc=vc-req;
stv=stv+req;
db=dc; // here the data buffer will be full anyway since it has stolen due to space
requirements
vsb=1
}
else
display error. //no space to steal.
}
}
else
if packets=audio
{
if((ac-ab)>=req)
{
ab=ab+req; //allocate the req size to the memory
}
else
drop packets;
}
else //video
{
if((vc-vb)>=req)
```

```
{
vb=vb+req; //allocate the req size to the memory
}
else
drop packets;
}
}
function output_handling()
{
if(output is from audio )
ab=ab-output.length;
else
if(output is from video)
vb=vb-output.length;
else
if(dc>db) // giving back data -- contains three cases.
{
free=dc-db;
if(free>dbb) //making sure the data memory size does not fall below default size.
free=free-dbb //while giving back the dc should not fall below dbb
if(asb==1 and vsb==0) // stolen from audio
{
if(free>sta)
{ac=ac+sta;
dc=dc-sta;
free=free-sta;
asb=0; sta=0
}
else
{
to_give=sta-free; //giving back only part of the memory
ac=ac+to_give;
dc=dc-to_give;
free=free-to_give;
sta=sta-to_give;
}
else //stolen from video
if(vsb==1 and asb==0)
{
if(free>stv)
{vc=vc+stv;
```

```
dc=dc-stv;
free=free-stv;
vsb=0; stv=0
}
else
{
to_give=stv-free;
ac=ac+to_give;
dc=dc-to_give;
free=free-to_give; stv=stv-to_give;
}
else
if(vsb==1 and asb==1) //stolen from video and audio
{
if(free>=stv+sta) // all the stolen space can be given back.
{
ac=ac+sta;
dc=dc-sta;
vc=vc+stv;
dc=dc-stv;
asb=vsb=0;
free=free-(sta+stv)
sta=stv=0;
}
else //part of the stolen space can be given back
{
//giving preference to audio
if(free>=sta)
{
ac=ac+sta; //giving back whole stolen part of audio back to audio buffer
dc=dc-sta;
free=free-sta;
sta=0;
asb=0;
vc=vc+free; //giving back remaining part of free space to video
dc=dc-free
stv=stv-free
free=0;
}}}}}}}
```

All the variables are initialised and their values are computed. Then each flow is identified by its common header and priority value. Based on the threshold values it will be decided as to how much of the flow can be forwarded for transmission and how many packets must be dropped.

4. Results and Analysis

The modification in algorithm design and traffic analysis has been carried out using the simulation tool C++ based software platform with video streaming environment attached to it [18] [19]. The simulated network is as shown in Fig. 2. The basic dumbbell topology was taken and three different sources were

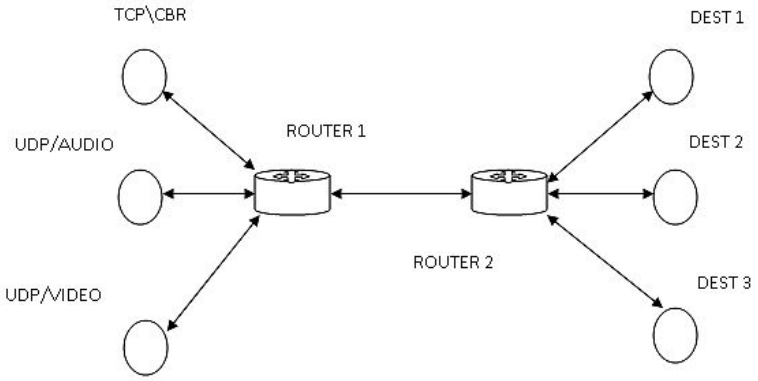

Fig 2. Simulated Network

connected to one end and send to the other end through two routers. The propagation delay throughout the simulation was set to 10ms for the entire network.

Network Parameters

The following statistics give a picture of the performance of the FMVC algorithm. The performance of FMVC algorithm with respect to network parameters such as delay, throughput and packet loss ratio has been analyzed. The throughput is the number of packets delivered fruitfully from one end to other end. Different flow combinations are taken and sent through the network and the throughput expressed as a percentage is illustrated in the Fig. 3(a).The total loss is the number of packet losses during the entire simulation run. As single flow goes

through the network there will not be any loss in the network, but when the flows increase there are packet loss as shown in Fig. 3(b). Just like in VC, the FMVC also does not contribute to the queuing delays. FMVC algorithm also helps the interleaving of packets from different flows and assures the throughput rate, but propagation delay experienced in the network needs to be considered. The average delay experienced by the flows for various bandwidths are illustrated in Fig. 3(c). Four different bandwidths are considered from 256 kbps to 8.1Mbps.

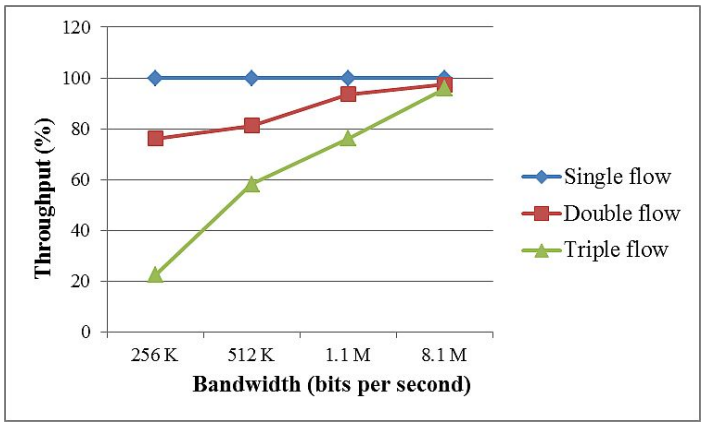

Fig. 3(a) .Variation of Throughput with bandwidth for various flows

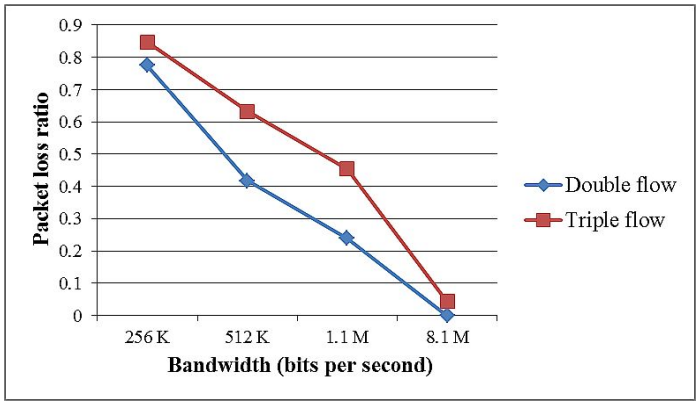

Fig. 3(b) .Variation of Packet Loss Ratio with bandwidth for various flows

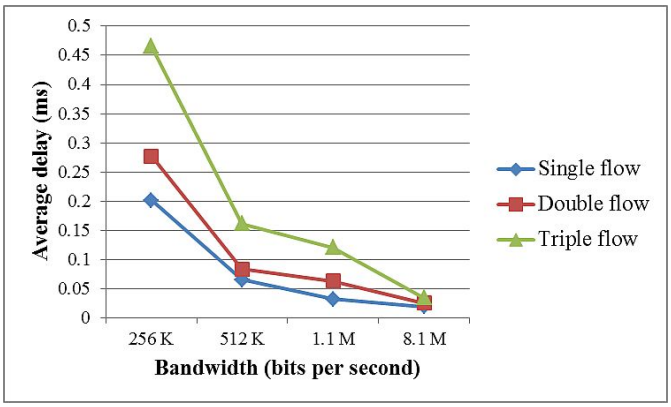

Fig. 3(c) .Variation of Average Delay with bandwidth for various flows

Variable Bandwidth Allocation

The goal of this test is to see how effectively the bandwidth allocation is carried out by the modified algorithm. The variable bandwidth allocation property of FMVC is analysed here with different combinations in the number of applications sent through the network. In the first case, when only a single application (CBR or Audio or Video) is allowed to pass through the network it is found that the bandwidth allocation is 100% as seen in Fig. 4(a). In the second case, when only two applications in different combinations are passed through the network the bandwidth allocation for each of the application is 50%. Fig. 4(b) depicts 50% bandwidth allocation for CBR and Video flows. Same property holds good for the other two combinations. The third case is allows all the three applications at once into the network. In Fig. 4(c), it is found that the bandwidth allocation is 50 % for video, 25 % for audio and 25 % for CBR as per the threshold limits set for each of the application.

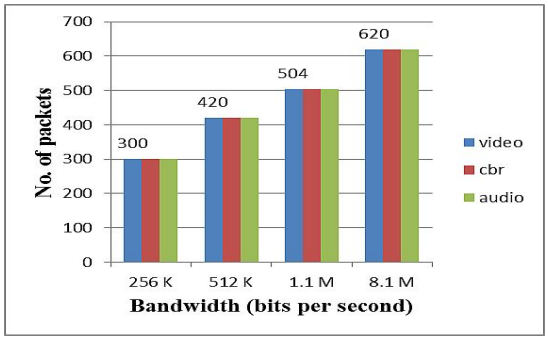

Fig. 4 (a). Packet allocation with bandwidth for Individual flows

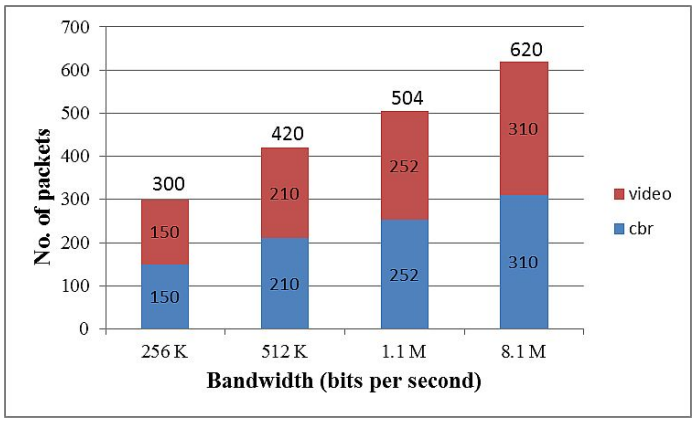

Fig. 4 (b). Packet allocation with bandwidth for Double flows

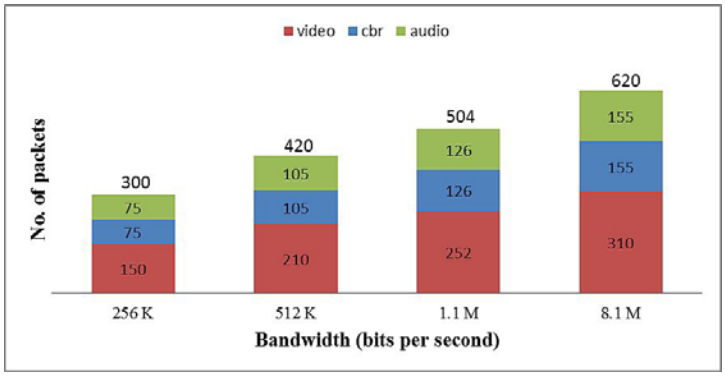

Fig. 4 (c). Packet allocation with bandwidth for Triple flows

Previous researchers has provided enough data on VC algorithm to prove its efficacy but stated its ineffectiveness in providing a fair share allocation for different types of flows. In case of FMVC, the throughput, average delay, packet loss ratio and the packet allocation for variable bandwidth indicates the effectiveness in providing the fair share allocation for various flows. To indicate the fairness an index has been used to compare the modified algorithm with the existing VC algorithm.

5. Fairness Index

With the available bandwidth, the algorithm must be fairer to all the flows traversing inside the router. An efficient algorithm does not necessarily mean that it is also fair. A single flow might take up the largest portion of the available bandwidth while the others remain idle. Noticeably, this is an adverse behavior

and in certain cases, gaining high fairness is valuable even at the cost of reduced efficiency. Instinctively, fairness is the closeness of the throughput achieved by each flow to its fair share [20]. To measure fairness, the Jain's fairness index formula is considered. The fairness index is defined as:

$$F(x) = \frac{(\sum x_i)^2}{n(\sum x_i^2)} \tag{1}$$

where, x_i is the throughput of the i^{th} flow and n is the total number of flows. The fairness index of a system ranges from 0 to 1, with 0 being totally unfair and 1 being totally fair. The fairness index for the two algorithms is shown in Table 1.

Table 1

Fairness Index

Bandwidth	VC	FMVC
8.1Mbps	0.54758	0.7954
1.1Mbps	0.4673	0.60114
512Kbps	0.2152	0.47371
256Kbps	0.10173	0.24715

6. Conclusion

This work describes the flow based modifications of virtual clock algorithm. The main idea is to provide fairness to the various scheduled flows by offering dynamic buffer allocation.

The performance of the modified algorithm was compared with VC. It is found that FMVC has better throughput than VC due to reservation for the TCP flows in the queue. It is observed that the drop in packets while simulating FMVC is less than that of VC for TCP flows in the network at the cost of UDP packet drop. The delay in packet delivery is analysed for FMVC and found to be less than that of VC. The bandwidth allocation property of FMVC is analysed. The observation shows that the video, audio and CBR take their fair share of the bandwidth. In the absence of one or two flows, the other flows occupy the unused bandwidth making the bandwidth allocation flexible. The fairness indexes for both algorithms indicate that FMVC is much fairer as compared to the VC algorithm. As a future work the FMVC algorithm can be implemented and tested in hardware. The various allocations for network traffic and the optimal network parameter values can be evaluated using real time data transmission via routers.

Acknowledgements

The authors would like to thank the unknown reviewers who reviewed this work. The authors would also like to thank Venkat Subramaniam of Mavjay solutions private limited, Chennai for giving suggestions during this research work.

R E F E R E N C E S

[1]. *J. Wang, Y. Li and H. Li*, A Novel Parallel Viterbi Decoding Scheme for NoC-Based Software-Defined Radio System, ETRI Journal, 35(2013), No. 5, 767-774.

[2]. *T. Faber*, "ACC: Active Congestion Control", IEEE Network, vol.12, no.3, (1998), pp. 61-65.

[3]. *M. Analoui , A. Gharegozi and Sh. Jamali*, "A new active mechanism to enhance the performance of the TCP congestion control", 2nd International Conference on Information and Knowledge Technology (IKT2005), Tehran, Iran, May 24-26, 2005.

[4]. *W. John and S. Tafvelin*, "Analysis of internet backbone traffic and header anomalies observed," in ACM IMC, 2007.

[5]. *W. John, S. Tafvelin, and T. Olovsson*, "Trends and differences in connection-behavior within classes of internet backbone traffic," in PAM, 2008.

[6]. *S Floyd and V Jacobson*, On Traffic phase Effects in Packet switched Gateways," Computer Communication Review, April 1991, (DOI 10.1145/122419.122421).

[7]. *A.Gharegozi*, "Experimental Evaluation of RED Queue Management Mechanism", International Conference on Intelligent Network and Computing (ICINC 2010), IEEE Computer Society and indexed by both EI (Compendex) and ISI Proceeding (ISTP), Kuala Lumpur, Malaysia, November 2010.

[8]. *D. Lin and R. Morris*, "Dynamics of random early detection," in Proc. of ACM SIGCOMM '97, Cannes, France, pp. 127-137, Oct. 1997, (DOI 10.1145/s263109.263154).

[9]. *L. Zhang*, "Virtual Clock: A new traffic algorithm for Packet Switching networks", Journal in Transactions on Computer Systems, vol. 9, no.2, (1998), pp 19-28.

[10]. *P.Sankar and C.Chellmuthu*, "Modified Virtual Clock Scheduling Algorithm for Queue Management", International Journal of Computer Theory and Engineering, U.P.B. Sci. Bull., Series C, vol. 75, Iss. 3, (2013), pp 43-54.

[11]. *A. Das, D. Dutta and A.Helmy*, Fair Stateless Aggregate traffic Marking using Active Queue Management Techniques, International Conference on Management of Multimedia Networks and Services: Management of Multimedia on the Internet , 211-223,2002.

[12]. *A. Rao, A. Legout, Y.sup Lim, D. Towsley, C. Barakat, and W. Dabbous*, Network characteristics of video streaming traffic, Proceedings of the Seventh Conference on emerging Networking Experiments and Technologies, December 06-09, Tokyo, Japan, (2011),1-12, (DOI 10.1145/2079296.2079321).

[13]. *D. Stiliadis*, "Latency – Rate Servers: A general model for analysis of traffic scheduling algorithms", IEEE International Journal on Networking, vol. 6, no.5, (1998), pp. 1-3.

[14]. *G. G. Xie and S. S. Lam*, 'Delay guarantee on Virtual Clock Server', IEEE Transactions on Networking, vol. 3, No. 6, (1995), pp. 1-5.

[15]. *G. Varghese, S. Suri and G.Chandranmenon*, Leap Forward Virtual Clock-a new fair queuing scheme with guaranteed delays and throughput fairness, PODC '97 Proceedings of the sixteenth annual ACM symposium on Principles of distributed computing ,(1997), 7-11 (DOI 10.1145/259380.259482).

[16]. *J. Chen, M. Devetsikiotis, C. Huang and I. Lambadaris,*Virtual Clock with Priority Buffer: A Resource Sharing Algorithm, in Proceedings of IEEE GLOBECOM 98, Sydney, Australia, November 1998.
[17]. *N. Alborz*, 'Implementation of Virtual Clock Scheduling Algorithm in OPNET', Thesis B.A., Shahid Beheshti University, Tehran, Iran, 1998.
[18]. *K. Fall and K. Varadhan*, 2005 the NS Manual.
[19]. http://www.isi.edu/nsnam/ns/ns2
[20]. *H. Natiq Jasem , Z. Ahmad Zukarnain ,and M. Othman*", Fairness of the TCP-based new AIMD Congestion Control Algorithm" Journal of Theoretical and Applied Information Technology, (2005),pp.568-57.