Asian Journal of Information Technology 5 (11): 1239-1244, 2006

A Scalable Jitter Management Scheme Through a Scheduler with Timestamps

[1]M. Usha and [2]R.S.D. Wahida Banu
[1]Department of Computer Science and Engineering,
Sona College of Technology (Affiliated to Anna University), Salem 636005, Tamil Nadu, India
[2]Department of Electronics and Communication Engineering, A.C. College of Technology,
Anna University, Karaikudi, Tamil Nadu, India

Abstract: Quality of Service and multimedia networking add new dimensions to the distributed multimedia applications. Approaches of congestion avoidance such as Random Early Detection favour TCP flows. Active queue management schemes like FRED support fairness, but, punish misbehaved, non-TCP flows. These mechanisms result in a significant performance drop in Quality of Service for multimedia flows because most of these flows use UDP. RED is the most popular AQM in IP routers since it reduces delay, compared to the traditional Drop Tail. Even though the delay is low, the delay variance or jitter is noticeable. Hence this work aims at addressing reduction of jitter while improving throughput. The design comprises a novel scheduler that works with an active queue management scheme. This scheduler has a single FIFO structure at the output queue of the IP router. This single FIFO removes the limitation in sorting complexity with the number of active flows. The proposed work uses per-packet time-stamping to enhance jitter management of multimedia flows. The ns-2 simulation results show that there is a significant reduction in jitter with the new scheduler.

Key words: Jitter, quality of service, scheduling algorithm, unified jitter management scheme, multimedia networking, time stamp

INTRODUCTION

As more and more organizations use the Internet for their multimedia applications, there is significant rise in the research interest on Quality of Service (QoS). Xiao *et al.*[1] provide an overview of QoS in the Internet. The various QoS parameters acceptable for applications are throughput, delay, delay variation (jitter), loss and error rates. For example IP telephony applications insist on bounded jitter.

The best-effort Internet model does not meet the growing QoS demands of business and user community applications. The IP routers play a major role in supporting QoS for such applications. Scheduling and Active Queue Management (AQM) are the two ways to support QoS in IP routers. AQM determines at the input queues which packets to be dropped. The scheduler at the output queue decides which packet to send next through the link[2]. The role of the schedulers is significant in enhancing the delay and jitter characteristics of multimedia applications including Video on Demand (VoD) and videoconferencing. AQM[1] can be classified into two classes.

- Rate-based AQMs which control the flow rate.
- Queue-based AQMs that monitor the queue at the congested link. Adaptive Virtual Queue (AVQ) uses virtual queues for congestion avoidance[3].

Random Early Detection, RED[4] comes under the queue-based scheme. RED had been recommended by IRTF at the IP routers[5]. Due to the characteristics of RED, TCP flows are benefited and UDP flows are punished. But due to its retransmission and acknowledgement techniques, TCP is not suitable for real time multimedia flows. User Datagram Protocol (UDP) is the preferred protocol for these[6]. Hence alternative solutions are needed to support distributed multimedia flows in the Internet.

The QoS constraints for these flows normally require throughput, delay and jitter as important parameters. Out of these, throughput and one of the others (delay, jitter etc.) is NP complete[7].

There had been many research works going on with delay as one of the QoS parameters[4,8-12]. Delay is not taken in this work as a QoS parameter. Though RED reduces the average delay of packets, it increases the jitter

Corresponding Author: M. Usha, Department of Computer Science and Engineering, Sona College of Technology (Affiliated to Anna University), Salem 636005, Tamil Nadu, India

of non-bursty (UDP) streams and hence their play out buffer requirements, thereby negating at least in part the gains on the lower mean delay. It is then expected that the audio quality perceived at the destination to be mediocre[13].

Another problem with RED is choosing appropriate RED parameters which are an open problem[14]. The authors have proposed an alternative two-class model with of a proactive AQM mechanism[15]. An improved version is Gentle Flow-based Proactive Queuing, GFPQ[16]. These two resulted in improved throughput. In this work the authors have taken jitter as the other parameter, since they are dealing with non-interactive multimedia applications where throughput and jitter are the two important QoS parameters
This paper focuses on:

- Link utilization
- Buffer utilization
- Less Jitter compared to FIFO and RED

The Contributions of this study are:

- Modified calendar scheduler for ns-2
- Modified real time scheduler for ns-2
- Better QoS through improved jitter and throughput for multimedia flows

RELATED WORK

This section presents a survey on relevant research. Earlier research classifies scheduling algorithms into those with and without jitter control[17]. If the scheduler does not consider jitter control it suffers from over allocation of buffer space at the receiver end. Jitter is reduced as the packets arrive at the receiver with consistent inter arrival times. Delay jitter is defined as the Standard Deviation of the delays experienced by packets traversing between two points[18].

Service provider could also provide better QoS as delay jitter is an important QoS parameter for non-interactive applications like VoD.

There is also a considerable amount of research work going on for delay only and loss only schedulers. They differentiate effectively only in delay and loss, respectively and the combined delay and loss scheduler differentiates effectively in both delay and loss[8]. Timestamp schedulers that consider reducing delay are: Weighted Fair Queuing[19], Worst-case Fair Weighted Fair Queuing (WF^2Q), Virtual clock[20] and self-clocked fair queuing[21]. Fair Priority Queuing (FPQ) algorithms approximate Generalized Processor Sharing (GPS) policy. The disadvantage of FPQ is that it needs per session basis buffering. WFQ has worst case virtual time complexity of GPS i.e., O(n). Algorithmic complexity of implementing a priority queue for N arbitrary numbers is O(log n). Both WFQ and WF^2Q maintain a reference GPS server. WFQ has a time complexity of O(n) which can be brought down to O(log n) in WF^2Q and Virtual Clock schedulers do not maintain any reference GPS server. Therefore, their time complexity is O (log N) per packet.

A family of scheduling protocols, called *Universal Timestamp-Scheduling*, is defined in[10] to forward packets in this network such that all members of the protocol family provide the same upper bound on packet delay as Virtual Clock scheduling. A *bounded-delay server* ensures that no packet along a channel will spend more than its delay bound in the node. Real-time packets, which are ineligible for transmission, are kept in a queue from which they are transferred to the scheduler as they become eligible.

This queue is maintained as a set of calendar queues[22]; packets are inserted in a queue indexed by their eligibility-time and all the packets that are in the queue indexed by the current time become eligible[9]. Stratified Round Robin[11] is a fair-queuing packet scheduler which has good fairness and delay properties and low quasi-O(1) complexity. This provides a good approximation of Weighted Fair Queuing and the scheduler, despite its low complexity, exhibits a constant single packet delay bound, independent of the number of flows. Some rate-guaranteeing schedulers assign timestamp to each incoming packet and queue the packets at the output queue in order of increasing timestamp. The packet with least timestamp is scheduled to the output link. Examples of theses schedulers are WFQ[19], Time shift scheduling[23] and Frame-based fair queuing[24].

This method differs in assigning timestamps as follows: The favoured flow's interarrival time at the time of enqueuing is taken as reference. At the scheduler, the jitter manager works in such a way that the local delay jitter bound is maintained for only one pair of packets at a time. Thus the implementation requires only soft states to be maintained.

This study combines the concept of time stamping with holding time. The jitter manager and scheduler work as follows:

The jitter manager does the regulating phase by delaying the packets that arrive early. This is done by the timestamp-based local delay introduction so that the packets arrive at the receiver with almost same delay variation. The scheduler is a simple FIFO scheduler that sends the packets as per their time stamp.

This study is simulated in ns-2[25] and evaluated on two schedulers: the calendar queue and the real time scheduler.

Calendar queue scheduler: Calendar scheduler[22] comprises of buckets that is an array of linked lists. Each list consists of the events scheduled and the time at which they have to be dequeued. The array consists of entries like bucket (0), bucket (1), etc., Each entry of the bucket array will point to a list consisting of events. The events will be inserted into the list based on the priority. While dequeuing, the list of events from bucket (0) is removed first and then bucket (1) and so on.

Real time scheduler: The real-time scheduler attempts to synchronize the execution of events with real-time. It is implemented as a subclass of the list scheduler in ns-2 . Real time scheduler delays the dequeue process for a certain amount of time and dispatched all the packets whose dispatch time is equal to or less than the current time.

PROPOSED UNIFIED SCHEME

This study now combines the AQM and the scheduler as a unified mechanism. RED[4] can act as the active queue management (AQM) for best effort Internet. TCP is not suitable for multimedia applications such as interactive and non-interactive video on demand and videoconferencing.

RED scheme starts dropping packets randomly before the buffer gets full. Thus, it forces the increase in the number of drops, which eventually reduces the throughput but avoids the congestion. With regard to the parameter called delay, there occurs large variation between the delay times of packets. This situation results in jitter, which reduces the overall performance and efficiency when considered for real time applications like multimedia.

The proposed scheduler works with the GFPQ AQM designed by the authors[15,16]. It had been shown that with the novel AQM mechanism, the throughput for the UDP flow could be improved while maintaining fairness to other flows. However, for non-interactive multimedia applications such as VoD, it is important to have reduced jitter also. This work aims at a novel scheduler at the IP router that gives better QoS through improved jitter. Performance evaluation for the new unified scheme is done and presented in a later section.

PROPOSED SCHEDULER

The packets of the favoured flow are given timestamps as they are enqueued. A non-interactive multimedia application's flow is taken as the favoured flow for which jitter is to be improved. The scheduler is modified in such a way that this flow gets the desired QoS. The jitter manager module is added to the original scheduler. This ensures the packet sequence as well as the constant delay jitter.

For the discussion here the definition of flow is taken from[26]. A flow r is an infinite sequence r0, r1, r2…, of nonnegative real numbers. Informally, each r_i represents the number of bits or packets that travel in flow at instant i.

The throughput is required to be bounded by the following:

$$\sum_{i=1}^{N} Ki \leq C \quad (1)$$

Where C is the bandwidth of the bottleneck link and K_i is the desired rate of the ith flow.

Jitter manager: The jitter manager module takes jitter control for our favoured flow. The interarrival time, IAT of two adjacent packets of the flow [t_1, t2] is found out.

$$Diff = t_2 - t_1 \quad (2)$$

Figure 1 certain threshold which is nothing but the desired jitter upper bound D_{UL}, the second packet is delayed by [D_{UL}-Diff]. Thus the delay is always maintained at this threshold. Wherever possible, this delay is maintained at most at the threshold value.

$$\left| TS_{PREV} - TS_{DCT} \right| \leq D_{UL} \quad (3)$$

Where

TS_{PREV}: Timestamp of the previous packet
TS_{DCT}: Timestamp of the current packet
D_{UL}: Upper bound of local delay jitter

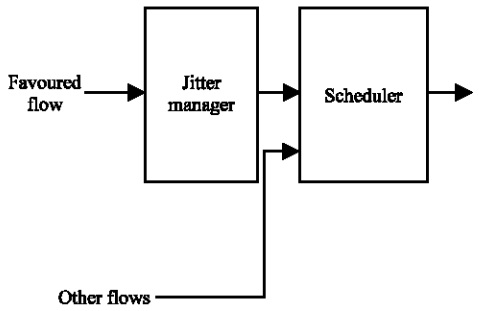

Fig. 1: Jitter manager in the jitter-guaranteeing times tamp scheduler JGTS

Because local jitter is bounded, end-to-end jitter is also bounded.

Jitter management algorithm: The algorithm for the jitter manager is presented below: The term current event's timestamp is calculated once it enters the incoming queue (enqueued). In default calendar scheduler, the events clock time refers to the sum of clock time and the line delay. It refers to the time at which the event is to be dispatched. That is, the events are scheduled in ascending order of event clock times in calendar scheduler.

If the flow is our favoured flow, then the current event's timestamp is compared with the previous one's timestamp. If the current one is less, then it means that it has entered the incoming queue long ago before the previous packet. So in order to ensure the constant delay jitter, the difference is found. If this difference value is less than bound then the current event is delayed by the (bound-difference) value. This means that the current event is delayed for just the constant delay time and allowed in. In the same way, the jitter manager works for the real time scheduler also.

At last the current event's timestamp and event clock time is stored for the purpose of comparing with the next arriving event. Ordering is important to reduce the time complexity involved in reordering at the receiver. The relative ordering of the packets is maintained in the IP routers themselves through the timestamps.

Time complexity: A small set of operational statements added to the existing scheduler does not affect the time complexity. Hence the time complexity of the new scheduler algorithm with jitter manager remains at O (1).

PERFORMANCE EVALUATION

This section presents the simulation results obtained through ns-2 simulator[8].

Simulation environment: The dumbbell topology is used in the simulation studies. The flow sources are named S_0 to S_N and destinations are D_0 to D_N. The sources send TCP and UDP traffic via two routers R_1 and R_2. The TCP sources use the Reno algorithm and UDP sources send CBR traffic.

The first router uses Drop Tail or one of the AQMs, RED or GFPQ. The simulation experiments were run with 10 flows introduced by the 10 nodes. Two of these are UDP out of which the second flow is taken as the favoured flow. The UDP send packets at a constant rates, which summed over all UDP sources equals 30% of the bottleneck speed. For scalability evaluation, the experiment was run with 100 flows out of which 20% are taken as multimedia flows using UDP. The link between R1 and R2 is taken as the bottleneck link with the line capacity C as 10 Mbps and 50 Mbps in the two sets of experiments.

Performance in terms of throughput: First the throughput is analyzed for the favoured flow in the case of RED scheme with calendar scheduler with that of real time scheduler.

Greater throughput is achieved for the favoured flow with the real time scheduler for a wired environment consisting of 10 nodes with the runtime of 1600 sec with $\rho = 2$. Moreover the scheduler part is modified in such a way that the favoured flow packets are allowed with a constant delay, which reduces jitter.

It is well known that throughput of the UDP flows suffer when RED is deployed as the AQM. However, with the modified real time scheduler, the throughput is seen to increase by 2%.

For better understanding of the evaluation, the Figures are drawn using exclusive tools. Fig. 2 shows the throughput of the multimedia flow for the '100flows' experiment.

Scalable jitter management: The delay jitter is measured by the following formula for Standard Deviation (SD):

$$S = \sqrt{\Sigma(X - M)^2 / n - 1}$$

Where

S = Standard deviation
X = Individual Interarrival Time (IAT)
M = Mean of all interarrival times
n = Sample size

The same experiment was now run for the AQM of RED at the input queue and the old and modified schedulers at the out put queue. The results presented depict four scenarios:

- Calendar scheduler+RED for 10 flows
- Calendar scheduler+RED for 100 flows
- Real time scheduler+RED for 10 flows
- Real time scheduler+RED for 100 flows

In all these cases, both the original and the new schedulers were tested.

Figure 3 shows that best performance improvement is seen with the real time scheduler, with reduction in SD being more than half. The simulation results show that the new scheduler exhibits linear behaviour. As the number of active flow increases, SD also increases.

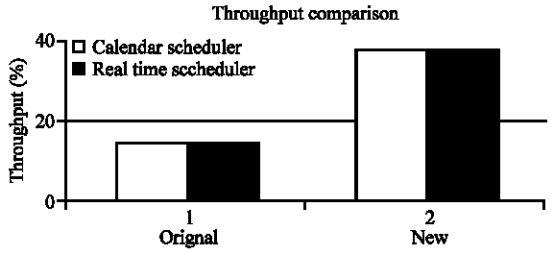

Fig. 2: Comparison of throughput for RED

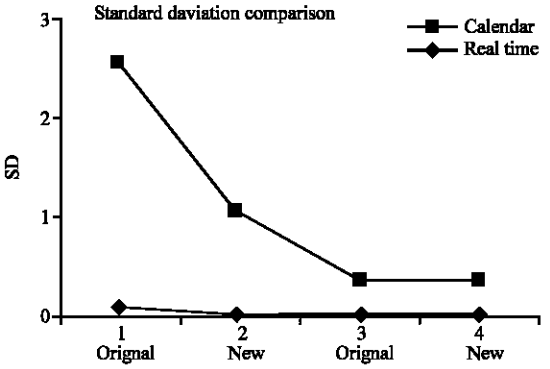

Fig. 3: Comparison standard deviation for RED

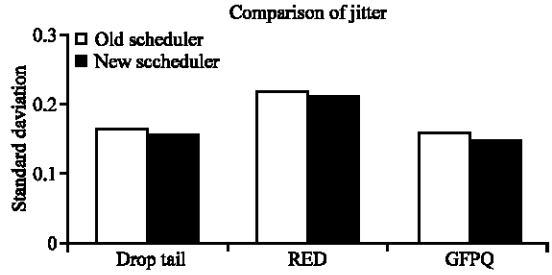

Fig. 4: Comparison of Jitter for the three schemes: DropTail, RED and GFPQ

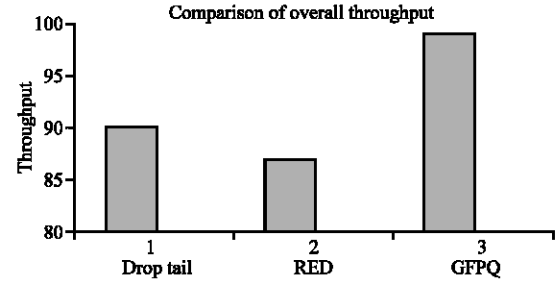

Fig. 5: Comparison of overall throughput with GFPQ and JGTS

However, with the new scheduler the value of SD is brought down. The second set of experiments was run with real time scheduler only as this gives a better jitter performance.

Unified jitter management scheme's performance: The previous set of experiments were performed with one AQM, namely RED and the old and new schedulers. The second set of experiments were conducted for the unified scheme tested with DropTail, RED and GFPQ.

Figure 4 shows that the SD is reduced for all the three schemes. The jitter reduced by 10% in all these cases. But the jitter reduction is prominent with GFPQ.

Impact on overall throughput: Figure 5 shows the comparison of throughput for the multimedia flows with the RED and GFPQ schemes along with the traditional Drop Tail queue. It is maximum for the GFPQ+new scheduler combination. The throughput of GFPQ is seen to improve by 12% compared to RED and by 9% compared to Drop Tail.

CONCLUSION

An unified jitter management scheme integrating an active queue management with a novel real time scheduler is designed and implemented. The new scheduler with a jitter manager is tested for performance in the wired environment and the results are compared with the original real time scheduler available with ns-2. It is observed that the jitter value has improved and weighed against the traditional schemes. This modified scheduler also increases the throughput value notably compared to the original one.

ACKNOWLEDGEMENT

We gratefully acknowledge Prof.B. Sathiyabhama and Prof. S. Jayabharathi of Sona College of Technology, Salem, India for their constructive comments that led to the refinement of this paper.

REFERENCES

1. Xiao, X. *et al*., 1999. Internet QoS: A Big Picture. IEEE Network, pp: 8-18
2. Keshav, S., 1977. Computer Networks: An Engineering approach. Addison Wesley, chapter 9.
3. Sasisankar, S. and R. Srikant, 2004. An adaptive virtual queue (AVQ) algorithm for active queue management IEEE/ACM trans. On Networking, 12: 397-413.
4. Floyd, S. and V. Jacobson, 1993. Random Early Detection gateways for congestion avoidance. IEEE/ACM Trans. On Networking, 1: 397-413.
5. Braden, *et al*., 1998. Recommendations on Queue Management and Congestion Avoidance in the Internet. RFC2309.

6. Jae, C. and M. Claypool, 2000. Better-Behaved, Better-Performing Multimedia Networking. SCS Euro media Conference.
7. Wang, Z. and J. Crowcroft, 1996. Quality-of-service routing for supporting multimedia applications. IEEE J. Selected Areas in Communications, 14: 1228-1234.
8. Striegel, A. and G. Manimaran, 2002. Packet Scheduling with Delay and Loss Differentiation. Computer Communications, 25: 21-31
9. Ferrai and Verma, 1990. A scheme for real time channel establishment in WAN. IEEE Jl. Of Selected Areas in Communication, 8: 368-379.
10. Jorge, C., 1999. Universal timestamp scheduling for real-time networks. Computer Networks, Elsevier, 31: 2341-2360.
11. Ramabhadran, N.S. and J. Pasquale, 2006. The stratified round robin scheduler: Design, Analysis and Implementation. IEEE/ACM Trans. Networking (to appear).
12. Goyal, P., S. Lam and H. Vin, 1995. Determining end-to-end delay bounds in heterogeneous Networks. Proc. of NOSSDAV.
13. Bonald, T., M. May and J. Bolot, 2000. Analytic evaluation of RED Performance. INFOCOM, pp: 1415-1424.
14. May, M., J. Bolot, C. Diot and B. Lyles, 1999. Reasons not to Deploy RED. Proceedings of 7th International Workshop on Quality of Service, pp: 260-262.
15. Usha, M. and R.S.D. Wahida Banu, 2005. Towards a Simple Service Class for Distributed Multimedia Applications in Next Generation IP Networks. Proc. of the ISCA 18th International Conference on Parallel and Distributed Computing Systems, pp: 161-166.
16. Usha, M. and R.S.D. Wahida Banu. A proactive mechanism for quality of service control in high speed networks. Intl. J. Business Inform. Sys., In Press.
17. Zhang, H., 1995. Providing end-to-end performance guarantees using non-work-conserving disciplines. Computer communications: Spl issue on system support for multimedia computing, pp: 18.
18. Yunkai, Z. and H. Sethu, 2000. A Simulation Study of Relationships Between Delay Jitter and Properties of a Switching Network. Proc. of the Applied Telecommunication Symposium.
19. Demers, A., S. Keshav and S. Shenker, 1990. Analysis and simulation of a fair queueing algorithm. J. Internetworking Res. Experience, 1: 3-26.
20. Zhang, L., 1990. Virtual Clock: A New Traffic Control Scheme for Packet Switching Networks, Proc. of ACM SIGCOMM '90.
21. Golestani, S., 1994. A Self-Clocked fair Queuing Scheme for Broadband Applications. Proc. Of IEEE INFOCOM.
22. Brown, R., 1988. Calendar queues: A fast O(1) priority queue implementation for the simulation event set problem. Comm. ACM, 31: 1220-1227.
23. Cobb, J., M. Gouda and A. El-Nagas, 1998. Time-shift scheduling: Fair scheduling of flows in high-speed networks. Proc of IEEE transanctions on Networking, 6: 274-285
24. Stiliadis, D. and A. Verma, 1996. Design and analysis of frame-based fair queuing: a new traffic scheduling algorithm for packet switched network. ACM SIGMETRICS.
25. NS2simulator. http://www.isi.edu/nsnam/ns
26. Jorge, C. and M. Gouda, 1997. Flow Theory'. IEEE/ACM Transactions on Networking, 5: 661-674.
27. Jon, C.R.B. and H. Zhang, 1996. {WF} 2 Q: Worst-Case Fair Weighted Fair Queueing. Proc. of INFOCOM, pp: 120-128.