Adaptive Early Random Drop: An algorithm for controlling queueing delay in a buffer

Mahmud Etbega Mohamed, M.E. Woodward

Abstract — **An adaptive version of Early Random Drop (AERD) is introduced to control the queueing delay in real-time multimedia applications to satisfy a user's quality of service requirements by dynamically moving a queue threshold. The aim is to keep queueing delay bounded as the arrival rate varies.**

Keywords — **QoS, queueing delay, Early Random Drop (ERD), queue threshold.**

I. INTRODUCTION

EARLY Random Drop (ERD) is a major modification of the Drop Tail (DT) algorithm and was designed to overcome the failures of DT as well as predicting congestion before it occurs. Basically ERD functions as a congestion avoidance mechanism instead of reaction after congestion occurs as in DT. The ERD mechanism consists of three basic steps: a) Predict congestion before it occurs. b) Identify the users contributing to this congestion. c) Signal these users to slow down [1]. ERD detects congestion when the instantaneous queue size reaches a drop level. This level indicates that a congestion period and we will feedback this information is fed back to the sources by dropping all new arriving packets with a fixed probability until the instantaneous queue size drops below the drop level [1] [2].

ERD has a better overview of the network traffic than DT, and has achieved a good overall network performance and provided a fair service to the network by reducing global synchronization in the network [1].

ERD succeeds in preventing the queue from overflowing by monitoring it constantly and this early detection improve fairness among network users by detecting aggressive users more accurately instead of warning all users to slow down. However the performance of ERD is still questionable as this success depends on the suitability of the Drop Level and Drop Probability to the current network traffic distribution [1].

In [1] Hesham suggested that improving the ERD algorithm to suit any network traffic could be achieved by implementing a dynamic adjustment of the Drop Level and the Drop Probability. Many research papers in the literature have already studied the development of dynamic adjustment of the drop probability such as RED [2], ARED [4]. These developments are dependent on the average queue size where the packet-marking probability varies linearly from 0 to maximum probability *maxp* as in RED [2] or they adjust the maximum packet dropping probability to adapt to changes in the mean queue length as in ARED [4], but none of theses methods specifically focuses on dynamically adjusting the drop level as traffic parameters change over time.

Mahmud E. Mohamed is a PhD student at the School of Informatics, University of Bradford, United Kingdom (e-mail: m.h.etbega@bradford.ac.uk).

Prof M.E Woodward is with School of Informatics, University of Bradford, United Kingdom (e-mail: m.e.woodward@bradford.ac.uk).

II. DESIGN ANLYSIS

A. Threshold position

The threshold L in the queue specifies the limit of the safe traffic area. Moreover choosing the threshold position is a very important design issue as it also specifies how much time is needed to signal the aggressive users to slow down. The aim of the system is to keep the end-to-end delay bounded at a specified value. Thus the threshold value should depend on the largest end-to-end delay of a user, but as there are multi users in one gateway and each of these user's has a different round-trip delay [1], so the threshold should be adjusted dynamically and this depends on the required delay in the gateway. It might argued that the threshold value should be set to a small value but that would not be possible simply because it might cause unnecessary packet losses.

B. Traffic source

It is very crucial to use a traffic model that emulates real network traffic in a realistic way. It is not be possible to test a system effectively without using a traffic source that represent and performs like real internet traffic. Therefore, a two state Markov Modulated Poisson Process (MMPP) traffic model was used to represent the underlying characteristics of a TCP flows. That is sudden switched in the arrival rate of a two-state MMPP source produce a similar effect to the sudden switches in arrival rate of a TCP source when congestion window is increased or decreased by a factor of two.

The MMPP-2 source is a Poisson process whose rate is a random process which varies according to an irreducible 2-state Markov chain. The MMPP is an extension of the Poisson process and is generally used as the input model of communication systems such as data traffic systems [3].

The MMPP traffic is fed in the queue with two different arrival rates. Each two state MMPP is representative of a TCP connection and consists of a sequence of data packets with different arrival rate.

C. Control mechanism

A continuous-time framework is used to model the queue and the time has been divided into time windows, where the round trip time RTT is less then one time-window.

Figure (1) shows the control feedback diagram. It is assumed that the RTT from the arrival process to queue and back to the arrival process through the delay controller is less the one time-window thus the arrival process can be considered to switch from one time-window to the next.

According to Guan[7] The following equation:

$$\overline{D_{k+1}} = 2D_r - D_k + G_{k-1} \quad (1)$$

Equation (1) is used to calculate the predicted mean delay in the next time window k+1 where D_r is the required delay which is a fixed value, D_k is the measured delay in the current time window and G_{k-1} is the measured delay error in the previous time window k-1.

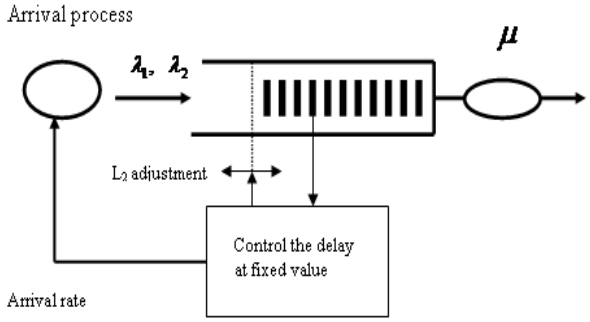

Fig. 1. Control system diagram

Once the predicted target mean delay in the next time-window k+1 has been calculated, the next step is to analyse the queue transition states to find the relation between the predicated target mean delay in the next time window and the position of the threshold for the next time-window L_{k+1} in order to achieve the predicted delay. In the next section a queueing analysis is carried out to find the drop value position L_{k+1}.

D. Queue analysis

The queue accommodates limited numbers of packets K; this number of packets includes any packet in the system server. The queue drop level or threshold is at position L. If the number of packets in the queue reaches the threshold, every new packet arriving to the queue is dropped with fixed probability. If the number of packets reaches the buffer size, all new arriving packets will be dropped and the source is signaled to stop sending packets by the feedback mechanism. Otherwise the source operates at normal sending rate [1]. The scheduling scheme of the queue is FCFS (First come first serve).

Figure (2) shows the arrival rate in each part of the queue when the arrival rate changes in the queue depending on the actual queue size.

As mentioned previously, an MMPP-2 is used for the arrival process and this MMPP-2 has two different arrival rates λ_1 and λ_2. The queueing model has three parts, the first part being when the number of packets is less then the drop level L in this part the arrival rate are λ_1 or λ_2 depends on the sending state. The effective arrival rate (arrival rate-drop rate) changes to $\lambda_1$or λ_2 when the number of packets reaches the drop line or threshold L and the dropping probability Pd will be used to randomly drop packets form the queue to signal the source to slow down. The last part is when the number of packets is larger than the queue capacity and so the effective arrival rate will be zero, where all packets will be dropped and the source will be signaled to stop sending [5][7].

Equations (2) and (3) show the dropping probability for state1 and state2 subsequently.

$$P_{d_1} = \begin{cases} 0 & q \le L \\ 1 - \dfrac{\lambda'_1}{\lambda_1} & L < q \le K \\ 1 & q \ge K \end{cases} \quad (2)$$

$$P_{d_2} = \begin{cases} 0 & q \le L \\ 1 - \dfrac{\lambda'_2}{\lambda_2} & L < q \le K \\ 1 & q \ge K \end{cases} \quad (3)$$

Again the mean queue length and the mean arrival rate are measured over each time window k, then these are used to compute the position of the threshold L for the next time-window $k+1$ in order to keep the delay at the required value [5][7].

The state diagram of the system is shown in figure (2). Since it has been assumed the mean time in each state of the arrival process is much larger than a time window, then this justifies the assumption that the system reaches a steady state before the next change in the arrival rate (at least on average) and we can thus model the system as a $M/M/1/L$ queue.

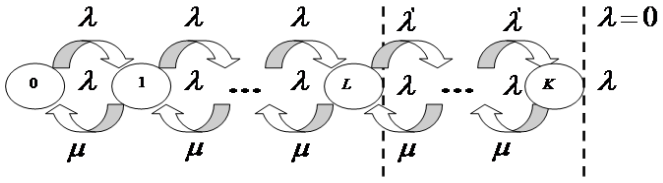

Fig. 2. Transition states

We measure the mean queue length (MQL) $\overline{x}(k)$ and the mean arrival rate $\lambda(k)$ over each time window k and their value is used to relocate the threshold for the next time-window k +1 to keep the delay at the required value.

In the case of $M/M/1/L$ queue, the balance equations are divided into three parts:

States 0, 1…. L

$$\lambda P(k) = \mu P(k+1) \quad (4)$$

Here k represents the states in the system from state 0 to state L.

$$P(k+1) = \frac{\lambda}{\mu} P(k) \quad k=0, 1, \ldots..L \quad (5)$$

The system is in equilibrium in this part of the queue and there are no dropped packets. The probabilities are as follow:

$$P(1) = \frac{\lambda}{\mu} P(0) \quad (6)$$

As the system is equilibrium the summation of the probabilities is equal to one. The general form of the probability is this part of the queue is:

$$P(k) = \rho^{k} P(0) \tag{7}$$

States L+1 .....K

In this part of the queue the there is a packet dropping probability in the system where the arrival rate changes from λ_1 or λ_2 to λ_1' or λ_2' depending on the arrival state. The state probabilities in this part of the queue are as follows:

$$\lambda' P(L) = \mu P(L+1) \tag{8}$$

$$\because\ P(L) = \frac{\lambda}{\mu} P(L-1) \tag{9}$$

The probability in state L is:

$$P(L) = \rho^{L} P(0) \tag{10}$$

$$P(L+1)\mu = P(L)\lambda' \tag{11}$$

And the probability in state $L+1$ is:

$$P(L+1) = \frac{\lambda'}{\mu} \rho^{L} P(0) \tag{12}$$

The general form for the probabilities in this part of the queue is:

$$P(L+i) = (\rho')^{i} (\rho)^{L} P(0),\ i=0,\ldots,K \tag{13}$$

Here i corresponds to the states in the system from state $L+1$ to the capacity of the system at state K.

$$\bar{x}(k) = \sum_{i=1}^{K} iP(i) \tag{14}$$

$$\bar{x}(k) = \sum_{i=0}^{L} i\rho^{i} P(0) + \sum_{i=L+1}^{K} i(\rho')^{i} \rho^{L} P(0) \tag{15}$$

Summing the series:

$$\sum_{i=0}^{L} i\rho^{i} = \frac{\rho}{(1-\rho)^{2}} \left[1 - \rho^{L} \left[1 + (1-\rho)L\right]\right] \tag{16}$$

Now let $i = L+1,\ldots\ldots K$ and let $j = i - (L+1),\ldots\ldots .K - (L+1)$. This makes $j = 0,\ldots\ldots K-(L+1)$ by using an appropriate summation of series:

$$\sum_{i=L+1}^{K} i(\rho')^{i} = \sum_{j=0}^{K-(L+1)} j(\rho')^{j} = \rho^{L} \frac{\rho'}{(1-\rho')^{2}} \left[1-(\rho')^{K-(L+1)}\left[1+(1-\rho')K-(L+1)\right]\right] \tag{17}$$

Using equations (16) and (17)

$$= P(0)\left[\frac{\rho}{(1-\rho)^{2}}\left[1-\rho^{L}\left[1+(1-\rho)L\right]\right]+\rho^{L}\frac{\rho'}{(1-\rho')^{2}}\left[1-(\rho')^{K-(L+1)}\left[1+(1-\rho')K-(L+1)\right]\right]\right] \tag{18}$$

Now we have to find the value of *P(0)* needs to be found.

$$P(0)\left[\sum_{i=1}^{L} \rho^{i} + \rho^{L} \sum_{i=L+1}^{K} (\rho')^{i}\right] = 1 \tag{19}$$

Then

$$P(0) = \frac{1}{\frac{1-\rho^{L+1}}{1-\rho} + \rho^{L} \frac{1-(\rho')^{K-L}}{1-\rho'}} \tag{20}$$

Both the mean queue length $\bar{x}(k)$ and the mean arrival rate $\lambda(k)$ are measured over each time window k in the simulation. The value of $\lambda(k)$ is used to calculate the threshold position for the next time window L_{k+1} to maintain the delay at the required value. By using Little's law the mean delay in each time window can be obtained as show in equation (21). After computing D_k, G_{k-1} and the required delay D_r is a constant, then these values can be used in equation (1) to find the value of the predicted mean delay in the next time window $k+1$. Once the predicated value for the mean delay in next time window $k+1$ is known the next step is to find a function for the threshold position L_{k+1} to maintain the mean delay in time window $k+1$ at the required value.

$$L_{k+1} = D_{k+1} \frac{\frac{\rho}{(1-\rho)^{2}}\left[1-\rho^{L}\left[1+(1-\rho)L\right]\right]+\rho^{L}\frac{\rho'}{(1-\rho')^{2}}\left[1-(\rho')^{K-(L+1)}\left[1+(1-\rho')K-(L+1)\right]\right]}{\left[\frac{1-\rho^{L+1}}{1-\rho}+\rho^{L}\frac{1-(\rho')^{K-L}}{1-\rho'}\right]\lambda(k)} \tag{21}$$

The bisection method is then used in our simulation to find the roots of equation (21) in order to find the threshold position for the next time window $k+1$ and subsequently adjust the threshold to the new position in an attempt to maintain the delay at the required value.

III. Simulation Results

A Java framework simulation is used in conjunction with the bisection method to find the roots of equation (21) to relocate the position of the threshold at the end of each time window to keep the delay in next time window at a specified value.

The parameters of the simulation are as shown below: The required delay (D_r)= 6ms, Arrival rate MMPP-2 state one (λ_1)= 0.2, Arrival rate in MMPP-2 State two(λ_2)=0.4, Transition rate in state one(γ_1)=0.01, Transition rate in state two(γ_2)=0.02, Time window length(TW)= 20ms, Dropping probability(P_d)=0.02, Service rate(μ)= 0.04 and Queue capacity (K)=15.

Hesham [1] suggested that the drop probability should be adjusted dynamically, depending on network traffic. However the purpose of this paper is to test the performance of the feedback mechanism rather than the performance of the entire network. Therefore we choose dropping probability 0.02 as recommended by Zhang in [6] although this version of Early Random Drop was not successful in controlling greedy users. We do not expect this modified version of early random Drop to solve all the problems of misbehaving users and other traffic problems but the goal of this Adaptive Early Random Drop (AERD) is to control the delay in the gateway.

The results of the above simulation are as shown in figure (3). The measured delay is fluctuates around the required delay as expected with variance (between measured and required delay) 0.965342. In the next step the time window length TW is changed in attempt to achieve a lower variance.

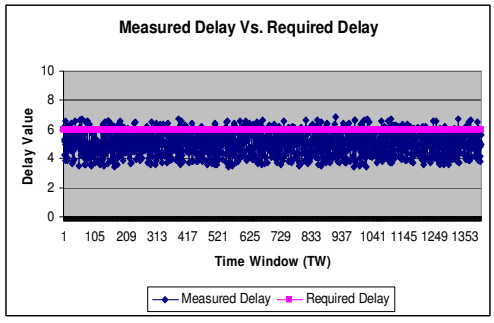

Fig. 3. Measured Delay vs. Required Delay at TW=20

Increasing the time window length to 25 gives the results in figure (4). The results have improved as the mean delay was maintained around the required delay and the variance is reduced to 0.847466.

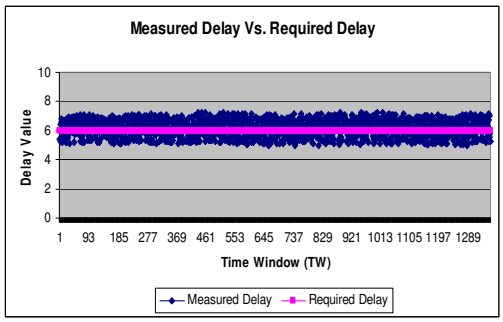

Fig. 4. Measured Delay vs. Required Delay at TW=25

From the results in figure (3) and (4) clearly the variance depends on the length of time-window, therefore it is crucial to find the optimum length of the time-window. In an attempt to obtain the optimum time-window the delay variance is plotted against time window length, as shown in Figure (5).

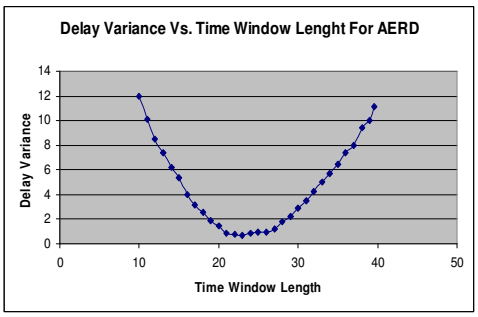

Fig. 5. Variance of Delay vs. Time window length

The measurement of the variance recorded high values at small sizes of time window then the value of variance decreases dramatically till it reaches the lowest value at time window length = 23, then it increases again.

In further attempt to lower the variance n previous time windows are used to predict the delay in next time window $k+1$. The number of time windows used is n= 2, 3, 4 .. and the variance in each case was obtained until the best value of the delay variance was achieved.

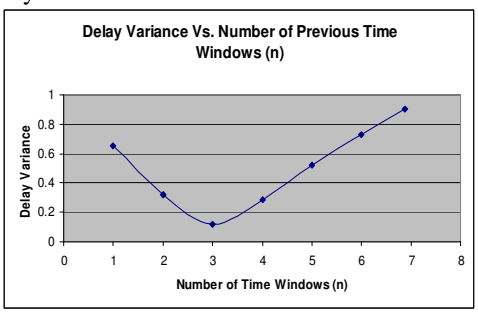

Fig. 6. Delay Variance Vs Different Number of Time Windows (*n)*

With n =2 the variance of delay is 0.324266, then when we increase the number of pervious time windows to three n=3 the variance has decreased to 0.12673. From that we could say that the variance decreased by increasing the number of the previous time windows n. However this is not always true as the delay variance has increased again with four previous time windows n=4. It is crucial to work out the optimum length of the time window therefore we plot a delay variance versus time length as illustrated in figure (6), which shows the lowest value of variance was at number of time windows n= 3.

IV. Conclusion

In this paper an adaptive version of Early Random Drop (AERD) has been introduced to maintain the mean queueing delay. The aim of this method is to satisfy the real-time multimedia application user's quality of service requirements.

The new AERD uses a dynamic threshold that can be adjusted with the aim of keeping the queueing delay in a finite buffer bounded at a specified value as the arrival rate changes over time.

The focus was on buffer delay which is a significant component of End-to-End Delay, and the disadvantages of the traditional Early Random Drop (ERD) have largely been eliminated by adding a dynamic threshold to the system instead of just using a static threshold value. An analytical model delay equation was also developed for an *M/M/1/L* queue. This equation has been used in the simulation in conjunction the bisection method to adjust the threshold to an effective value to maintain the delay around a specified value.

We have noticed that the accuracy of the results depends critically on accurate measurements of arrival rate. Optimum values have been obtained for time window length and number of time windows to use in the delay measurements for the parameters used, although further investigations would be required to determine the generality of these results if the system parameters (arrival rates and service rate) are changed.

References

[1] Hashem, E., "Analysis of random drop for gateway congestion control", *Report LCS TR-465*, Laboratory for Computer Science, MIT, Cambridge, MA, 1989, p.103.

[2] Van Jacobson and Michael Karels. Congestion avoidance and control. In Proceedings of ACM SIGCOMM, pages 314–329, 1988.

[3] Anderson, A. T. and Nielsen, B. F, "A markovian approach for modeling traffic with long-range dependence", IEEE Journal on Selected areas in communications,1998, Vol. 16(5), pp. 719-732.

[4] S. Floyd, G. Ramakrishna, and S. Shenker. Adaptive RED: An algorithm for increasing the robustness of RED's active queue management. Technical Report, 2001.

[5] Mahmud Etbega-Mohamed, M.E. Woodward "Adaptive Drop Tail: An Algorithm for maintaining delay bounds in a buffer with time-varying arrival rate". Proceedings of ICCTA'08 IEEE, Damascus-Syria, 2008, pp 625-226.

[6] Zhang, L., "A New Architecture for Packet Switching Network Protocols", MIT/LCS/TR-455, Laboratory for Computer Science, Massachusetts Institute of Technology, August 1989.

[7] Guan, L. M. E. Woodward and I. U. Awan, "Bounding Delay through a Buffer using Dynamic Queue Thresholds". In Proceedings of AINA, 2006, Pp. 623-628