International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:5, 2008

Adaptive Early Packet Discarding Policy Based on Two Traffic Classes

Rawya Rizk, Rehab Abdel-Kader, and Rabab Ramadan

Abstract**—**Unlike the best effort service provided by the internet today, next-generation wireless networks will support real-time applications. This paper proposes an adaptive early packet discard (AEPD) policy to improve the performance of the real time TCP traffic over ATM networks and avoid the fragmentation problem. Three main aspects are incorporated in the proposed policy. First, providing quality-of-service (QoS) guaranteed for real-time applications by implementing a priority scheduling. Second, resolving the partially corrupted packets problem by differentiating the buffered cells of one packet from another. Third, adapting a threshold dynamically using Fuzzy logic based on the traffic behavior to maintain a high throughput under a variety of load conditions. The simulation is run for two priority classes of the input traffic: real time and non-real time classes. Simulation results show that the proposed AEPD policy improves throughput and fairness over that using static threshold under the same traffic conditions.

Keywords**—**Early packet discard, Fuzzy logic, packet dropping policies, quality-of-service (QoS), TCP over ATM

I. Introduction

RECENTLY broadband wireless networks have been designed to support a variety of service types. This makes Asynchronous Transfer Mode (ATM) technology suitable for internet traffic. Several mechanisms have been presented to improve multimedia transmission over wireless networks. These mechanisms can be mainly categorized as either packet discard policies or buffer management policies.

A major objective of the packet discard policies is to minimize the packet fragmentation problem that arises due to discarding the whole packet even if it only contains a small part of corrupted data. The packet discard policies can be classified into *Partial Packet Discard* (PPD) [3] and *Early Packet Discard* (EPD) [4]. The EPD policy is shown to be fairer than the PPD and provides minimum bandwidth for each flow. In PPD, when a cell has been discarded, all subsequent cells belonging to the same packet are discarded. This results in significant waste in bandwidth and network resources while delivering the leading cells of the packet, which were stored in the buffer prior to the losses. The EPD attempts to save this bandwidth by rejecting an entire packet if upon arrival of its first cell, the queue exceeds a threshold value. Several modifications to EPD have also been proposed that address fairness and throughput issues [5-10]. The on-demand packet discard (ODPD) policy [10] avoids the packet fragmentation problem by dropping the cells that belong to full packet if the buffer becomes full.

The unpredictable and bursty traffic nature for ATM makes it is necessary to manage the buffer in order to implement different traffic classes. Buffer management is a fundamental technology to provide quality of service (QoS) control mechanisms, which control the assignment of buffer resources among different flows and flow aggregations according to certain policies. Limited researches were done considering the QoS of TCP traffic over ATM [11-13]. A bounded-delay type buffer is presented in [11]. It considers only the timeout problem by dropping packets when they exceed a maximum delay. In [12], the set of performance requirements are specified to the algorithm as a set of per-class QoS constraints. These constraints can be any mix of relative and absolute constraints. In [13], a partial push-out mechanism based on two thresholds is presented. In [14], The ODPD policy is modified to implement two priority classes.

The buffer management using Fuzzy logic approach is suggested in some work related to internet [15-18]. The use of Fuzzy logic can provide a robust mathematical framework for dealing with imprecision since it has a greater ability to adapt to dynamic, imprecise, and bursty environment.

In this paper, an adaptive early packet discard (AEPD) policy for two traffic classes is proposed. It comprises a priority scheduling and a fragmentation scheduling. The priority scheduling improves the performance of real time TCP traffic over ATM networks. It adjusts a dynamic threshold based on the traffic load changes of both real-time traffic and best-effort traffic. The fragmentation scheduling avoids the partially corrupted packets problem by differentiating between the cells of one packet from another. Since, the ATM cell header contains a cell loss priority (CLP) bit; the proposed policy checks this bit at each arriving cell before arranging the cell in the appropriate waiting queue.

The rest of the paper is organized as follows: Section II presents the TCP traffic classes over an ATM networks and introduces the proposed adaptive early packet discard (AEPD) policy. Section III shows the operation of the proposed AEPD using an example. The simulation environment and the simulation results are presented in Sections IV and V; respectively. Finally, Section VI summarizes the main conclusions.

R. Y. Rizk is with the Electrical Engineering Department, Faculty of Engineering, Suez Canal University, Port-Said, Egypt (*Corresponding author ; e-mail: r.rizk@ scuegypt.edu.eg).

R. F. Abdel-Kader is with the Electrical Engineering Department, Faculty of Engineering, Suez Canal University, Port-Said, Egypt (e-mail: r.abdelkader@scuegypt.edu.eg)

R. M. Ramadan is with the Electrical Engineering Department, Faculty of Engineering, Suez Canal University, Port-Said, Egypt (e-mail: rabab.ramadan58@gmail.com).

II. Adaptive Early Packet Discard Policy

Providing QoS guarantees are critical for real-time applications over internet [19]. In this paper, in order to accommodate the QoS requirements of various traffic types, the traffic is classified as either *real-time services* (Class 1) or *best-effort services* (Class 2). The QoS class required by each application is part of the contract negotiation procedure between the user and the network. In the ATM switch, the cell loss priority (CLP) bit in the ATM cell header is used to determine either the cell is high-priority (CLP = 0) or low-priority (CLP = 1).

The AEPD policy comprises two main functions: priority scheduling and fragmentation scheduling.

1) The priority scheduling guarantees the QoS requirements for Class 1 traffic by using a threshold that controls the access to the buffer for incoming traffic. If the buffer occupancy is below threshold level, it allows both types of traffic and if the occupancy level is more than threshold, it accepts only Class 1 traffic.
2) The fragmentation scheduling differentiates the buffered cells of one packet from another in order to avoid dropping cells from multiple packets, which causes the network to be underutilized.

A. Priority Scheduling

While priority mechanisms achieve low loss and delay for the higher priority class, they cause performance degradation for the lower priority class. In many cases, the buffer space reserved for the higher priority class is unused and wasted. The choice of threshold value has a great effect on the performance. If the threshold is low, it yields enough space only for Class 1 to be accommodated in the buffer at the expense of Class 2 increasing the loss of Class 2. On the other hand, high threshold value results in an increased loss of Class 1. The proposed AEPD adjusts the threshold as the traffic load of the two priority classes changes. It uses an adaptive threshold controller that determines the threshold value T based on the cell arrival rates of the two priority classes.

It is assumed that, each arriving cell is a cell of Class I (i.e., a high-priority cell) with probability R_I, or a cell of Class II (i.e., a Low-priority cell) with probability $R_{II} = (1- R_I)$. Thus, the arrival rates of the cells of the two classes are $\lambda_I = R_I \lambda$ and $\lambda_{II} = R_{II} \lambda$, where λ is the total load.

The adaptive threshold controller uses R_I and R_{II} as the input linguistic variables and T as the output linguistic variable. The terms “Low”, “Medium”, and “High” are used to describe both R_I and R_{II}. Thus, the term set for R_I is defined as T(R_I) = {Low (L_1), Medium (M_1), High (H_1)}. Similarly, the term set for R_{II} is defined as T(R_{II}) = {Low (L_2), Medium (M_2), High (H_2)}. On the other hand, the term sets “Small”, “Medium”, “Big” are used to describe T. Thus the term set for T is defined as T(T) = {Small (S), Medium (M), Big (B)}.

The membership functions of input and output linguistic variables are assumed to be trapezoidal membership functions since it is suitable for real-time applications [20]. The degree of membership of each trapezoidal function (f) will be described by:

$$\mu = f(x ; a, b, c) \tag{1}$$

where x is the linguistic variable and a, b, and c are slope, flatness and center of trapezoidal; respectively.

The universe of discourse of the two probabilities R_I and R_{II} are as usual ranges between 0 and 1. Assuming the buffer size is K, the universe of discourse of T ranges between 0 to K. The membership functions for T(R_I), T(R_{II}), and T(T) are defined as follows.

$$\mu_{L_1}(R_I) = f(R_I;\ 1/0.2,\ 0.2,\ 0) \tag{2}$$
$$\mu_{M_1}(R_I) = f(R_I;\ 1/0.2,\ 0.2,\ 0.5) \tag{3}$$
$$\mu_{H_1}(R_I) = f(R_I;\ 1/0.2,\ 0.2,\ 1) \tag{4}$$
$$\mu_{L_2}(R_{II}) = f(R_{II};\ 1/0.2,\ 0.2,\ 0) \tag{5}$$
$$\mu_{M_2}(R_{II}) = f(R_{II};\ 1/0.2,\ 0.2,\ 0.5) \tag{6}$$
$$\mu_{H_2}(R_{II}) = f(R_{II};\ 1/0.2,\ 0.2,1) \tag{7}$$
$$\mu_{S}(T) = f(T;\ 1/(0.25 \times K),\ 0.25 \times K,\ 0) \tag{8}$$
$$\mu_{M}(T) = f(T;\ 1/(0.25 \times K),\ 0.2 \times K,\ \tfrac{1}{2} \times K) \tag{9}$$
$$\mu_{B}(T) = f(T;\ 1/(0.25 \times K),\ 0.25 \times K,\ K) \tag{10}$$

Table I describes the rule structure for control of the threshold T. As can be seen in Table I, only 6 rules appear from the 9 possible rules in the knowledge base of the fuzzy threshold controller. The remaining three are not included as they would never be activated. For example, if R_I is H_1 that means R_I is in the range between 0.7 and 1, which also means that R_{II} can not exceed 0.3, i.e., R_{II} can not be in M_2, or H_2.

TABLE I
RULE STRUCTURE FOR CONTROL OF T

R_I / R_{II}	L_1	M_1	H_1
L_2	M	S	S
M_2	B	M	
H_2	B		

The adaptive threshold controller assumes Larsen product inference method for the inference engine and Center of Area (CoA) defuzzification method for the defuzzifier.

B. Fragmentation Scheduling

The buffer is organized to differentiate between partial packets and full packets. The AEPD policy drops from the full packets first, if necessary to avoid the fragmentation problem. Fig. 1 shows the buffer arrangement. It has the following portions: a transmit queue (TQ), and four packet waiting queues: the full high-priority packet queue (FH), the full low-priority packet queue (FL), the partial high-priority packet queue (PH), and the partial low-priority packet queue (PL). The ATM switch will only transmit the packets in TQ, while the packets in the waiting queues must be transferred first to the transmit queue before transmission.

The policy checks the CLP bit at each arriving cell before arranging this cell in the appropriate waiting queue. Both FH and PH contain Class 1 cells, while FL and PL contain the cells of Class 2. The connection of any arrived cell is tagged as one of four states: receiving, transmitting, drop_whole, or drop_partial. The switch processes the cell according to the state of the associated tag. If the tag state is drop_whole, the whole packet including the end of packet (EOP) cell will be discarded. If the tag state is drop_partial, non of the EOP cells are dropped and the EOP cell will be kept.

The overall operation of the proposed AEPD policy consists of three main parts: the *Cell process*, the *Drop packet*, and the *Cell scheduler*.

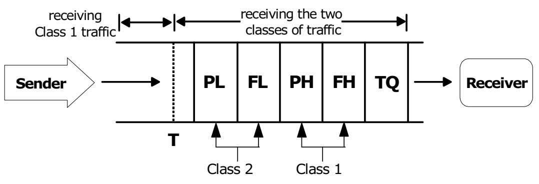


Fig. 1 Buffer Arrangement

– *Cell process:*

The cell process depends on the position of the incoming cell in the packet. Fig. 2 presents the pseudo code of the Cell process. It includes three possible states: first cell of the packet, intermediate cells and EOP cell.

When *the first cell* of a packet of connection *i* arrives at the switch, the tag state is assigned to receiving. Then the policy checks the CLP field in the header of the ATM cell. If the CLP field is '1' the cell is appended to PL, else it is appended to PH.

If the arriving cell is an *intermediate cell* in the packet (i.e., it is not the first cell or the EOP cell), the switch processes the cell according to the state of the associated tag. If the state of the tag is drop_partial or drop_whole, then the incoming cell is discarded; otherwise, the cell will be buffered. If the buffer is not full and the tag state is receiving, the cell is appended to PL or PH according to whether the CLP = 1 or 0; respectively. If the buffer is not full and the tag state is transmitting, the cell is appended to TQ. If the buffer is full, the algorithm will select a packet to be discarded to free a space of the buffer. Then, it calls the Drop packet process.

When *the EOP cell* of connection *i* arrives at the switch, the cell will be dropped if the tag state is drop_whole; otherwise the cell will be buffered. If the buffer is not full and the tag state is receiving, the cell is appended to PL and the packet will be transferred from PL to FL or the cell is appended to PH and the packet will be transferred from PH to FH according to whether the CLP = 1 or 0; respectively. If the buffer is not full and the state is transmitting or drop_partial, the cell is appended to TQ. The EOP is dropped only if buffer overflow occurs and there are no packets in the waiting queues. If this happens, the algorithm will call the Drop_packet process.

```
If C is the first cell of the packet and the buffer not full Then
    Tag state = receiving.
    If CLP=1 Then
        If K < T, Then append the cell to PL
        Else discard the cell
        End If
    Else append the cell to PH
    End If
Else If C is the first cell of the packet and the buffer full Then
    If CLP=1 Then
        Discard the cell.
        Tag state = drop_whole.
    Else Call Drop_packet
        append the cell to PH.
Else If C is the EOP cell Then
    If tag state "drop_whole" Then
        discard the cell and free the tag state
    Else If buffer not full Then
        If the tag state is receiving Then
        If CLP=1 Then
            append the cell to PL and move the packet from PL to FL
        Else append cell to PH and move the packet from PH to FH
        End If
        Else append the cell to TQ and free the tag_state.
        End If
    Else (the buffer full)
        Call Drop_packet
        If the buffer is still full or the tag state is drop_whole Then
            discard the cell
            If the tag state is drop_whole Then
                free the tag state
            Else tag state = drop_partial
            End If
        End If
    End If
Else If the cell is an intermediate cell
    If the tag state is drop_whole or drop_partial Then
        discard the cell
    Else If the buffer is not full Then
        If the tag state is receiving Then
            If CLP=1, Then append the cell to PL
            Else append the cell to PH
            End If
        Else append the cell to TQ
        End If
    Else (the buffer is full)
        Call Drop packet
        If the buffer is still full or the tag state is drop_whole Then
            discard the cell
            If the tag state is drop_whole Then
                free the tag state
            Else tag state = drop_partial
        End If
    End If
End If
```

Fig. 2 Pseudo Code of Cell Process

– *Drop packet:*

The Drop packet process is presented in Fig. 3. The AEPD policy drops by default the front packet of FL. If FL is empty, then it drops the front packet of PL. If PL is empty, then it drops the front packet of FH. If FH is empty, then it drops the front packet of PH. If all the waiting queues are empty, the AEPD policy acts as PPD policy. Then, the incoming cell is discarded and the tag state is changed to drop_partial.

```
If FL is not empty Then drop the front packet in it
Else If PL is not empty Then drop the front packet in it
Else If FH is not empty Then drop the front packet in it
Else If PH is not empty Then drop the front packet in it
End If
```

Fig. 3 Pseudo Code of Drop packet

– *Cell scheduler:*

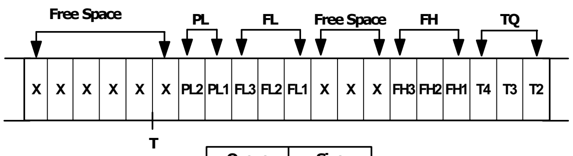
Fig. 4 shows the pseudo code of the cell scheduler. If the transmit queue (TQ) is not empty, the AEPD policy will transmit the first cell in it. If the length of the TQ is smaller than a predefined threshold (T_q), then the policy checks the four waiting queues. If FH is not empty, move the front packet in it to TQ. Else, it checks by order the PH, the FL, and finally the PL. The tag state will be free if the packet is moved from the full packet queues, FH or FL; otherwise, the tag state will be transmitting.

```
If the transmit queue TQ is not empty Then
    transmit the front cell in it.
End If
If TQ less than the threshold $T_q$ Then
    If FH is not empty Then
        move the front packet from FH to TQ and free the tag_state.
    Else if PH is not empty Then
        move the front packet from PH to TQ and set the tag_state to transmitting.
    Else if the FL is not empty Then
        move the front packet from FL to TQ and free the tag_state.
    Else if PL is not empty Then
        move the front packet from PL to TQ and set the tag_state to transmitting.
    End If
Else transmit the front cell in TQ
End If
```

Fig. 4 Pseudo Code of Cell Scheduler

III. THE OPERATION OF THE AEPD

The following example illustrates the operation of the AEPD policy. It is assumed that there are four input connections (Conn1, Conn2, Conn3, and Conn4). Conn2 and Conn3 send high priority data while Conn1 and Conn4 send low priority data. The processing time is 10 times more than the scheduling time, this means that each 10 cells are sent to the buffer, only one cell is transmitted from it. For simplicity, it is assumed that the arrival rate of Class1 is about the same of the arrival rate of Class 2 and the threshold is static. Assuming, buffer size = 20 cells, buffer Threshold T = 15 cells (75 % of the buffer size), and transmit queue threshold T_q = 4 cells (20% of the buffer size). Each packet consists of three cells. At the first time slot, Each of Conn2, Conn3, and Conn4 has one packet (A3, A2, A1), (B3, B2, B1), and (C3, C2, C1); respectively. Conn1 has the last cell of a packet (PL3). The current state of the buffer is arranged as follow: TQ contains 3 cells, FH contains 3 cells, FL contains 3 cells, PL contains 2 cells, and free buffer space equal to 9 cells. Fig. 5 shows this arrangement.



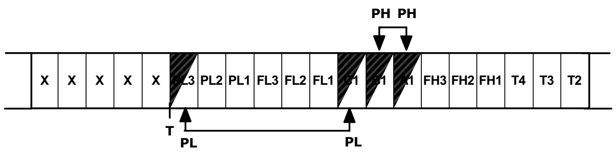
Queue	Size
TQ	3
FH	3
PH	empty
FL	3
PL	2
Free Space	9

Fig. 5 The initial buffer arrangement

A. The cell process:

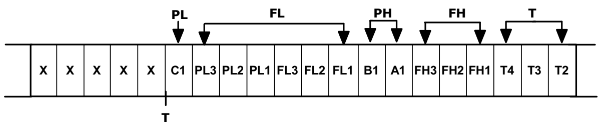
– *During the processing time while the queue length does not exceed the threshold.*

The algorithm processes each incoming cell and puts it in the appropriate place of the buffer. Each accepted high priority cell is appended to PH, and each accepted low priority cell is appended to PL. When a packet in PH or in PL queue is complete, the algorithm moves the packet from this queue to FH or to FL; respectively. Fig. 6(a) shows the buffer state after accepting two Class 1 cells (A1 and B1) and two Class 2 cells (C1 and PL3). Then, the packet that includes (PL1, PL2, and PL3) became full packet. Fig. 6(b) shows the same state after moving this full packet from PL to FL.


PH PH
X X X X X PL3 PL2 PL1 FL3 FL2 FL1 C1 B1 A1 FH3 FH2 FH1 T4 T3 T2
T
PL
PL

Queue	Size
TQ	3
FH	3
PH	2
FL	3
PL	4
Free Space	5

(a)


PL
FL
PH
FH
T
X X X X X C1 PL3 PL2 PL1 FL3 FL2 FL1 B1 A1 FH3 FH2 FH1 T4 T3 T2
T

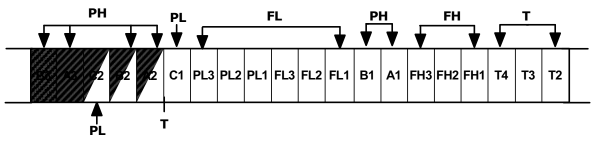
Queue	Size
TQ	3
FH	3
PH	2
FL	6
PL	1
Free Space	5

(b)

Fig. 6 (a) The buffer state after processing the incoming cells while the queue length does not exceed the threshold
(b) The rearrangement of the current buffer state

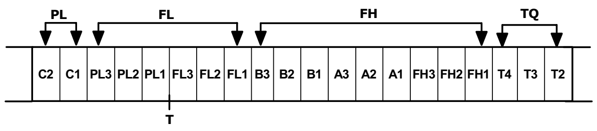
– *During the processing time after the queue length exceeds the threshold.*

After the queue length exceeds the predefined threshold, the policy checks the CLP bit for each incoming cell. It discards the cell if it is the first cell of a packet and its CLP=1 (new Class 2 packet). Else, the cell is accepted while there is a space in the buffer. Since the incoming cells, A2 and B2 are Class 1 cells, they are appended to the appropriate waiting queue in the buffer. The incoming cell C2 is appended to the buffer because the preceding cells of its packet are already in the buffer. The cells A3 and B3 are accepted. Then, the buffer is full. Fig. 7(a) shows the buffer state in this case. Fig. 7(b) shows the buffer state after the complete packets are moved from PH to FH.



Queue	Size
TQ	3
FH	3
PH	6
FL	6
PL	2
Free Space	empty

(a)



Queue	Size
TQ	3
FH	9
PH	empty
FL	6
PL	2
Free Space	empty

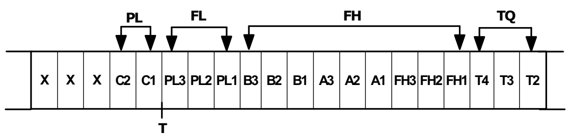
(b)

Fig. 7 (a) The state when the buffer became full
(b) The rearrangement of the current buffer state

It is clear from Fig. 7 that, after receiving A1 and A2 cells the algorithm appends them to PH. After receiving A3 (EOP cell) at PH, the whole packet is moved from PH to FH. Also after receiving PL3 (EOP cell), the completely related packet is moved from PL to FL. The buffer now is full but the algorithm is still in the processing state. The algorithm in this stage needs to drop a packet to set a free space on the buffer. According to the conditions of dropping the packet, the dropped packet will be the front packet in FL. Then, the algorithm will be ready to accept the new receiving cells.

– *During the processing time after dropping the packet.*

Fig. 8 shows the buffer state in this case. Dropping the packet leads to that there will be a space for three new cells. Then, the new cell C3 is accepted and appended to PL. Because C3 is EOP cell, then the whole packet related to this cell is moved from PL to FL. At this time, the processing time ends and the scheduler time starts.

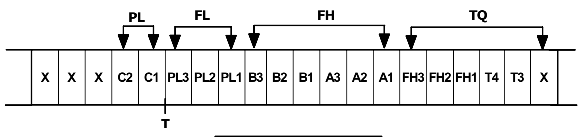


Queue	Size
TQ	3
FH	9
PH	empty
FL	3
PL	2
Free Space	3

Fig. 8 The buffer state after dropping the packet

B. The cell scheduler:

The algorithm first sends the front packet in TQ and sets its place free. Then it checks if the transmit queue length is less than the transmit queue threshold or not. In this state TQ=3 cells and T_q = 4 cells. This means that the transmit queue length is less than the threshold and it can accept a packet from the packet waiting queues. According to the conditions of accepting this packet, the accepted packet will be the front packet in FH. The algorithm in this case moves the front packet from FH to TQ. Fig. 9 shows the buffer state in this stage.



Queue	Size
TQ	5
FH	6
PH	empty
FL	3
PL	2
Free Space	4

Fig. 9 The buffer state after the cell scheduler process

IV. Simulation Environment

In the simulation, a simple topology is chosen to study the performance dynamics. The TCP connections are connected to the inputs of the ATM switch. For simplicity, all the link delays from the switch to the sources and receivers are assumed to be the same. Also, all the links have a capacity of 155 Mbps. Since the TCP packet size often used in IP networks is 512 bytes, then the TCP data packet plus 40-byte TCP/IP header is segmented into 12 ATM cells. The ATM switch operates in a slotted manner. The simulation is performed using MATLAB and is run for 10^5 time slots.

The traffic can be basically classified into five categories: data, voice, video, images, and graphics [21]. It is assumed that these categories are confined to two types of sources that represent the two predefined priority classes:

1) *Poisson arrival* streams are generated by generating the independent and identical exponentially distributed

interarrival times. They are mostly used to characterize the pattern of arrivals input traffic [9,21]. The probability of *X* arrivals is:

$$\Pr[X=i]=\begin{cases}\dfrac{\lambda^i \ \exp(-\lambda)}{i!} & for \ \ i \geq 0 \\ 0 & for \ \ i<0\end{cases} \tag{11}$$

where λ is the arrival rate. Although, the Poisson model breaks down in a number of real-time applications such as voice, it is still an extremely useful model when dealing with data and relatively new sources such as packetized images [17].

2) The *ON/OFF* or *bursty* source model is widely used in the simulation of ATM switches [22]. A bursty source alternates periods of activity in which it transmits at peak cell rate with periods of silence in which it does not transmit. This source model is very suitable for the representation of packetized voice, and interactive data services.

   In the simulation, it is assumed the *ON/OFF sources* with the number of cells per burst has a geometric distribution with a mean of *E*[*x*]=5 cells, the duration of the idle phase has an exponential distribution with a mean of *E*[*s*]=0.14772 sec., and the intercell time during a burst is t_c=0.016 sec. In order to obtain the performance measures versus the cell rate (load), it is assumed that the cell rate varies due to a change in the average number of cells per burst, while the average silence time is assumed to be constant (*E*[*s*]=0.14772 sec.).

Multiplexing the two traffic sources explained above is an important step in examining the behavior of the system. It is assumed that the *ON/OFF* sources represent Class 1 traffic with CLP = 0, while the Poisson processes represent Class 2 traffic with CLP =1.

The default values of the system parameters are listed in Table II.

TABLE II
LIST OF PARAMETERS

Parameter	***Symbol***	***Value***
Number of TCP connections	*N*	32
Buffer size	*K*	1000 cells
Traffic load	λ	0.6
Threshold for the buffer	*T*	75 % of *K*
Threshold for transmit queue	T_q	20 % of *K*
Throughput	*G*	
Fairness index	*F*	

Three performance metrics are evaluated, loss, throughput, and fairness. The *throughput* is the average number of packets delivered by the network per time slot. It can be derived from:

$$G=\frac{Number \ \ of \ \ deliverd \ \ packets}{Total \ \ number \ \ of \ \ packets} \tag{12}$$

The fairness can be intended in terms of throughput of each connection. The system is fair if the throughput of each connection is approximately equal. The Fairness always lies between 0 and 1 and thus a higher fairness index indicates better fairness between connections. Given a set of throughputs of *N* connections *(G_1, G_2, ..., G_N)*, the fairness index is calculated as follows [2]:

$$F(G_1,G_2,...,G_N)=\frac{(\sum_{i=1}^{N} G_i)^2}{N\sum_{i=1}^{N} G_i^2} \tag{13}$$

V. Simulation Results

Figures 10-13 show the comparison between the static threshold queue (Fixed Threshold), the adaptive threshold queue (Dynamic Threshold) and the policy without implementing priority. Fig. 10 gives the percentage loss versus load. The results show that the loss of Class 1 for Dynamic Threshold is the lowest among all curves. For high load ratios, the performance of Dynamic Threshold is better, where as for low values there is no significant difference between Class 1 in the two policies (Dynamic and Fixed thresholds). It is clear from the figure that the percentage loss of Class 1 for Dynamic Threshold doesn't exceed 7% at moderate load (0.6), as opposed to 8% for Class 1 at Fixed Threshold and 13% for the policy without implementing priority. The loss of Class 2 is highly minimized in the case of Dynamic threshold compared to Fixed threshold. The difference between the loss of Class 2 at the two schemes reaches more than 10% at high loads. This is due to the fact that, on high load ratios; more of the buffer space is allocated to Class 1 at the cost of Class 2. Fig. 11 shows loss against buffer size at load=0.6. As expected, the loss decreases with the increase of the buffer size. It is clear that, Class 1 at both Dynamic and Fixed thresholds has lower loss at different buffer sizes.

Fig. 12 presents the throughput versus load. It is clear that, the throughput of Class 1 at Dynamic threshold reaches more than 90% as opposed to 87% for Class 1 at Fixed threshold and 74% for the policy without thresholds. Fig. 13 shows that the throughput increases with the increase of the buffer size. Dynamic Threshold policy achieves throughput 92% for Class 1 and 74% for Class 2 as opposed to 90% and 58% for Class 1 and 2 for Fixed Threshold; respectively.

Fig. 14 presents the fairness in both Fixed Threshold and Dynamic Threshold. The fairness index tends to be 1 at high load at Dynamic threshold. The difference between the two policies decreases as the load increases. This is due to that, at high load the buffer is always full by the two types of traffic and there is no unused buffer space.

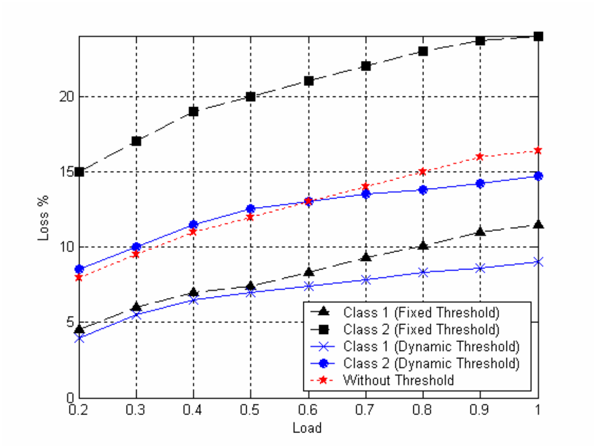


Fig. 10. Loss percentage versus load at buffer size =2000 cells.

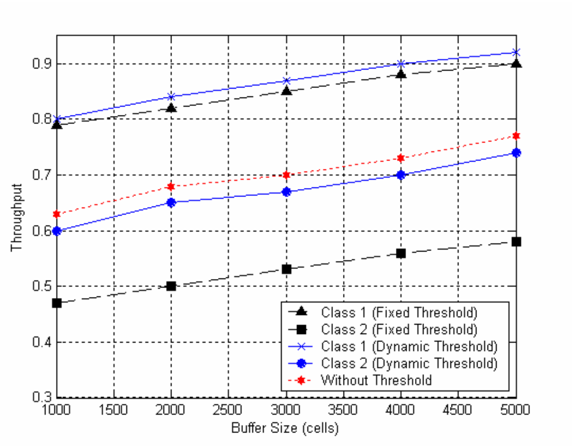


Fig. 13. Throughput with respect to buffer size at load = 0.6.

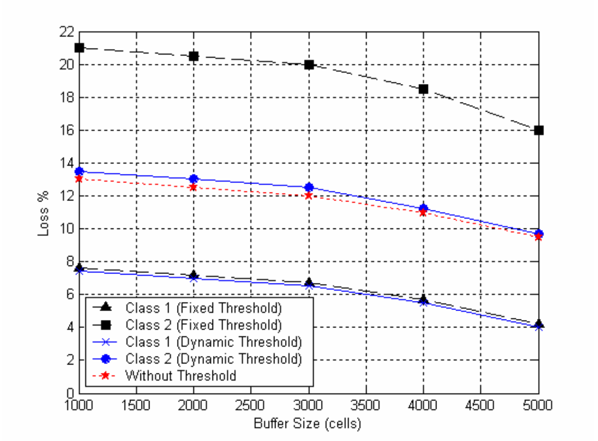


Fig. 11. Loss percentage versus buffer size at load =0.6.

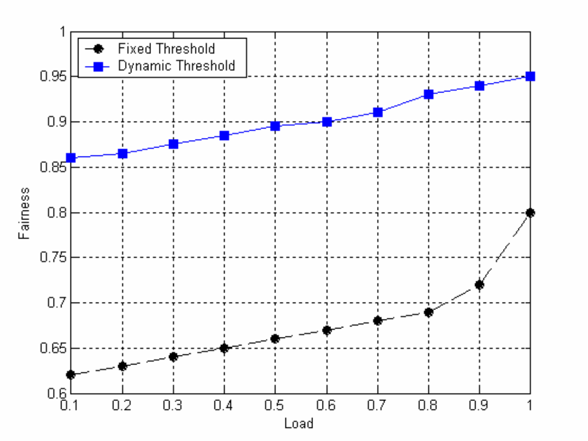


Fig. 14. Fairness against load.

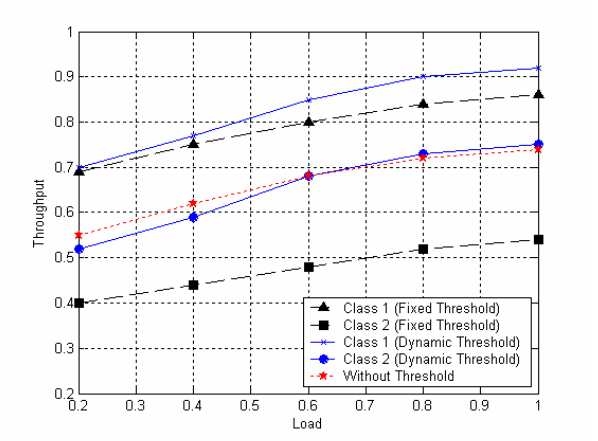


Fig. 12. Throughput versus load with buffer size =2000 cells.

VI. Conclusion

In this paper, an adaptive early packet discard (AEPD) policy is proposed to overcome the poor performance of TCP traffic over ATM. The proposed policy improves the performance of real time traffic and avoids the fragmentation problem. Two classes of services are considered that represent real-time and best effort traffic. The policy uses complete buffer sharing. It differentiates between the two traffic classes and differentiates the buffered cells of one packet from another. Simulation results show that the proposed AEPD policy achieves an improvement in loss percentage and throughput reaches in some cases more than 10% compared to both the traditional packet discard policy without implementing priority and the priority policy using static threshold. The proposed AEPD achieves fairness and guarantees the QoS requirements of real time TCP traffic over ATM networks.

References

[1] O. Bonaventure, and J. Nelissen, “Guaranteed frame rate: a better service for TCP/IP in ATM networks,” *IEEE Network*, pp. 46-54, January/February 2001.

[2] A. Ishtiaq, Y. Okabe, and M. Kanazawa, "Management of parallel UBR flows over TCP in congested ATM networks," *ELSEVIER Computer Communications*, vol.27, pp.801-808, 2004.
[3] J. A. Grenville, and M. A. Keith, "Packet reassembly during cell loss, *IEEE Network*, vol.7 no. 5, pp. 26-34, 1993.
[4] S. Floyd, and A. Romanov, "Dynamics of TCP traffic over ATM networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 4, pp. 633-641, 1995.
[5] M. A. Labrador, and S. Banerjee, "Packet dropping policies for ATM and IP networks," *IEEE Communications*, vol. 2, no. 3, pp. 2-12, 1999.
[6] W. K. Lai, and C. C. Liu, "SWFA: A new buffer management mechanism for TCP over ATM-GFR," *IEEE Transactions on Communications*, vol. 51, no. 3, pp. 356-358, March 2003.
[7] Z. Lotker, and B. Patt-Shamir, "Nearly optimal FIFO buffer management for two packet classes," *Computer Networks*, vol. 42, pp. 481-492, 2003.
[8] P. Dube, and E. Altman, "Goodput analysis of a fluid queue with selective discarding and a responsive bursty sources," *IEEE INFOCOM*, 2003.
[9] P. Dube, and E. Altman, "Queueing and fluid analysis of partial message discarding policy," *Queueing Systems*, vol. 44, pp. 253-280, 2003.
[10] D. Pao, "On-demand packet discard scheme for TCP over ATM-UBR service," *IEE Proceeding-Communication*, vol. 151, no. 3, pp. 190-196, June 2004.
[11] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, and M. Sviridenko, "Buffer overflow management in QoS switches," *Society for Industrial and Applied Mathematics (SIAM)*, vol. 33, no. 3, pp. 563-583, 2004.
[12] J. Liebeherr, and N. Christin, "JoBS: Joint buffer management and scheduling for differentiated services," in *Proc. of the 9th International Workshop on Quality of Service*, pp. 404-418, 2001.
[13] V. Hristov, and F. Ibrahim, "Data performance of modified pushout + PPD mechanism with elastic and deterministic traffic sources with priorities," in *Proc. ComSysTech'06*, 2006.
[14] W. Saber, R. Rizk, and A. Sallam, "Prioritized on-demand packet discard scheme for TCP over ATM," *Port-Said Engineering Research Journal*, vol. 11, no. 2, pp. 1-10, September 2007.
[15] S. Kausha, and R. Sharma, "Modeling and analysis of adaptive buffer sharing scheme for consecutive packet loss reduction in broadband networks," *International Journal of Computer Systems Science and Engineering*, vol. 4, no. 1, pp. 8-15, 2007.
[16] K. Shihab, "Performance tuning of novell netware based on Fuzzy reasoning," *International Journal of Computers*, issue 1, vol. 2, pp. 80-86 ,2008.
[17] S. T. Zargar, and M. H. Yaghmaee, "Fuzzy green: a modified TCP equation-based active queue management using Fuzzy logic approach, *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 6, no. 5A, pp.50-58 , May 2006.
[18] M. H. Yaghmaee, and G. K. Tousi, "Fuzzy jobs: a Fuzzy extension to the jobs algorithm," *IAENG International Journal of Computer Science*, vol. 34, no. 1, 2007.
[19] H. Iiang, W. Zhuang, X. Shen, and Q. Bi, "Quality-of-service provisioning and efficient resource utilization in CDMA cellular communications," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 1, pp. 4-15, January 2006.
[20] S. Nascimento, B. Mirkin, and F. Moura-Pires, "Modeling proportional membership in Fuzzy clustering," *IEEE Transactions on Fuzzy Systems*, vol.11, no. 2, pp. 173-186, 2003.
[21] S. Prahmkaew, and C. Jittawiriy anukoon, "Performance evaluation of adaptive rate control (ARC) for burst traffics over ATM network," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 6, no. 1B, pp. 78-84, January 2006.
[22] M. Schwartz, *Broadband Integrated Networks*. Upper Saddle River, NJ: Prentice-Hall, 1996.