Efficient QoS Support of Continuous Media Stream Packets with Variable Size and Diverse Deadlines*

Zoe Antoniou & Ioannis Stavrakakis
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115

ABSTRACT

Real time applications require the timely delivery of application-level data units (ADUs) which may consist of one or more packets. As a result, it would be reasonable to associate a *common* deadline with all the packets in one data unit.

A packet based scheduling policy is proposed and investigated in this paper. The policy is shown to be optimal in the sense that it maximises the number of complete packets *scheduled* for transmission at any point in time, by taking into consideration the associated diverse deadlines and the packet sizes.

I. INTRODUCTION

ITU-T has proposed the Asynchronous Transfer Mode (ATM) as the emerging technique for efficient switching and multiplexing of future multimedia services. Real time applications are expected to share a large portion of the traffic handled by ATM networks. The traffic characteristics of real time applications can be very diverse as they combine a variety of media data which, in general, have stringent requirements in terms of delay, loss and jitter. Typical such applications are digital video and audio [1].

Consider, for instance, a video application which sends pictures (frames) to a network at a rate of 30 frames per second. Roughly every 33.3 ms, the video application reads and packetises the picture, and sends packets to the network. Bits generated over a frame form a logical block of information (application data unit or ADU) which could provide a basis for the description of the generated traffic as well as the desirable QoS. In this environment, frames are typically generated at a constant frame rate and, thus, the nature of such an application is periodic.

An application data unit can be mapped into one or more packets and each packet consists of a collection of cells. In order to guarantee a timely reconstruction at the receiver, complete frames should be delivered at a constant display rate of 30 frames per second. This implies that all cells of an application data unit should be delivered in time for the reconstruction process in order to avoid either starvation or buffer overflow at the receiving end [2], [3].

When resources are over-allocated to a number of applications -to increase network utilisation- all cells of an application data unit may not be delivered before this application data unit is required by the reconstruction process. The required QoS for such applications may then be defined in terms of a metric associated with the correctly and timely delivered application data units. An application-level QoS metric such as the application data unit loss rate would describe more accurately the QoS delivered to the end user, than a network-level QoS metric such as the cell loss rate.

The requirement for timely delivery of information associated with such real time, continuous media traffic suggests that deadlines could be defined and deadline-driven policies be designed. If the nature of an application requires that a complete application data unit should be received before it can be processed by the receiver, then it would be reasonable to associate a common deadline with all the cells in the same data unit and develop scheduling policies that aim to maximise the number of complete data units that are delivered by their respective due date. This approach is different from many traditional approaches in which every cell is assigned its own deadline and the scheduling policy typically attempts to maximise the number of individual cells transmitted by their due date [4], [5].

Notice that the size of the periodically arriving ADU is variable and that ADUs of different streams are allowed to have different deadlines. In this paper it is demonstrated that policies such as the Earliest Due First (EDF) and the Shortest Job First (SJF) are suboptimal in the environment considered here (variable size ADUs which have diverse deadlines) and a different approach is proposed.

The proposed scheduling policy aims to maximise the number of application data units *served* by their deadlines from all data streams by maximising the number of packets *scheduled* for complete service by their respective deadlines. In order to achieve this objective, the proposed policy makes scheduling decisions based on the length of the packets as well as their relative deadlines. It is argued and illustrated that the

*Research supported by the National Science Foundation under Grant NCR 9628116.

proposed optimal scheduling leads to efficient actual service which under certain arrival processes is shown to be the optimal.

II. SYSTEM DESCRIPTION AND DISCUSSION OF SOME EXISTING SCHEDULING POLICIES

In this paper, a network node with N incoming and one outgoing links is considered. Each link corresponds to one (continuous media) application. All links are assumed to have the same capacity and, thus, a cell requires a fixed amount of time for transmission referred to as a slot. All arrivals and departures from the node occur at slot boundaries. Let J denote an incoming data stream (application); J takes values in the range $0 \leq j \leq N-1$. A variable with superscript j represents a quantity associated with stream j.

The per stream (application) arrival process is assumed to be bursty, delivering application data units (ADUs) at random arrival instants. Each ADU consists of a batch of variable size packets, measured in cells, and can tolerate a maximum delay (queuing plus transmission) at the node, referred to as the deadline, D^j. The Quality of Service (QoS) of these applications is assumed to be directly impacted by the number of complete packets received by the end user. Thus, the QoS metric considered here is the packet loss probability. A packet is assumed to be lost if any one of its cells is lost.

Some related existing policies are briefly summarised below in an attempt to illustrate their inefficiencies and motivate the proposed policy. More detailed descriptions and explainations can be found in [6].

The **Earliest Due First** (EDF) scheduling policy has been widely considered and studied in the past. It has been shown that the EDF policy is optimal, in the sense that it maximises the number of fixed-sized information units served before their respective deadline. But it can be shown that it will fail to maximise the number of complete packets (blocks of variable number of cells) served by their respective deadline.

The **Shortest Job** (packet) **First** (SJF) policy is known to maximise the number of packets served over any time interval, when packets do not expire (no dealdines). When packets do have deadlines the SJF policy will not necessarily maximise the number of (complete) packets served.

The EDF and SJF policies either focus on the deadline of a packet or its size in order to make scheduling decisions. There exist realisations for which these policies fail to maximise the number of complete packets served by their respective deadlines. The policy proposed in the next section schedules and serves packets using the basic principles of both the EDF and the SJF policies. It makes decisions based on both the size of the packets and their relative deadlines.

III. THE PROPOSED SD SCHEDULING POLICY

The proposed policy is described in this section and will be referred to as the **Size-Deadline**-based (SD) scheduling policy. The SD policy aims to maximise the number of unexpired complete packets that can be served over any time interval by taking into consideration both the deadlines and the sizes of the packets. Although the SD policy may not maximise the number of unexpired packets served for certain realisations, it can be shown that it has a number of properties which establish its potentially very high effectiveness or prove its optimality under certain arrival processes.

The SD policy is executed upon ADU arrival. At this time, all packets present in the system are scheduled for transmission. New scheduling decisions are taken for packets which arrived before the current ADU arrival time instant and they are not fully transmitted. The system may be viewed as consisting of one arrival process delivering ADUs belonging in different classes (with different dealdines) or, equivalently, as consisting of multiple streams each with its own deadline. The latter model is adopted in this paper.

As it will become clear below, the SD scheduling policy is executed in two conceptually distinct stages. In the first the *scheduling* algorithm is executed. The *service* algorithm is executed in the second phase. These algorithms are described in the next subsections and they are executed every time an ADU arrives (decision times).

A. *The scheduling algorithm*

The scheduling algorithm is the central component of the proposed SD scheduling policy and determines which packets will be scheduled for service and which ones will be dropped at the current decision time. The packets scheduled (for service) will be actually served completely and before the expiration of their respective deadlines provided that no future arrivals interfere. If some of the scheduled packets are in the system in the next decision (ADU arrival) instant, some of these packets may actually be dropped at that time, since all packets are re-scheduled.

If t denotes the arrival instant of a stream j ADU of size P, then this ADU must be served over any P slots within the interval $[t, t+D^j]$. Since these slots are shared among all streams, taking away P slots to serve this ADU will potentially impact on the number of packets from other streams which can be scheduled. It is easy to see that the impact is minimised if this packet is scheduled over the interval $[t+D^j-P, t+D^j]$, that is over the last P eligible slots. In this case, the earlier slots in $[t, t+D^j-P]$ become available to other streams whose packets may expire earlier. Notice that the actual service algorithm (and not the scheduling algorithm) will determine the actual service order.

Let t denote the current decision (arrival) time instant. Let $\mathcal{P}_t = \{P_1, P_2, ..., P_n\}$ denote the set of packets in the system at time t, ordered according to increasing size or remaining service time; let $\hat{P}_i$ denote the size of P_i. Let $\mathcal{S}_t = \{T_1, T_2, ..., T_k\}$ denote the set of the (remaining) delay tolerances of the packets in $\mathcal{P}_t$, assuming that $T_1 < T_2 < ... < T_k$. Notice that $k \leq n$ and $T_i \leq max_j\{D^j\}$. Let $\mathcal{T}_i = \{P_l : \text{delay tolerance of } P_l \text{ is equal to } T_i\}$. Let I_i be a counter associated with $\mathcal{T}_i$ whose (initial) value at t is equal to T_i. The scheduling algorithm is described next.

The scheduling algorithm considers the shortest packet first (P_i) and examines whether it can be served within its delay tolerance (step 3-(a)). If not, it considers the next shortest packet. Otherwise, over the next $\hat{P}_i$ iterations it reduces the delay tolerances (counters) whose value in the current iteration is higher or equal to this packet's current delay tolerance (step 3-(b)-(i)); it does not modify the remaining delay tolerances (counters) (step 3-(b)-(ii)). Finally, the current value of the counters is kept to be used as its initial value when the next packet is considered (step 4).

At the arrival (decision) time t:

1. Create $\mathcal{P}_t$; that is order the packets in the system according to SJF. Create $\mathcal{T}_i$. Initialise counters I_i, $1 \leq i \leq k$.
2. Select the shortest packet in $\mathcal{P}_t$, say P_i, and remove it from $\mathcal{P}_t$.
3. Suppose that $P_i \in \mathcal{T}_i$ then:
   (a) If $I_i - \hat{P}_i < 0$, drop P_i and repeat step 2.
   (b) Else, set $I_l(0) = I_l$ for $1 \leq l \leq k$; set $m = 1$ and repeat for $\hat{P}_i$ steps:
      i. $I_l(m+1) = I_l(m) - 1$, if $I_l(m) \geq I_i(m)$.
      ii. $I_l(m+1) = I_l(m)$, if $I_l < I_i(m)$.
4. Set $I_l = I_l(\hat{P}_i)$.

Notice that if the counter associated with the currently considered packet, say $\tilde{I}$, has a value in the current iteration (step 3-(b), for some m) that is higher than that of some counter I, it is implied that the current packet is scheduled (so far) over slots beyond the deadline associated with counter I and, thus, no reduction in the delay tolerance of such packets should be considered (step 3-(b)-(ii)). If I has a value equal to or higher than $\tilde{I}$ in the current iteration, its value should be reduced by one (as for $\tilde{I}$), since the slot represented by the current iteration is legitimate for service of packets with the delay tolerance indicated by I.

B. The service algorithm

The scheduling algorithm described above identifies the maximum number of packets which can be served before their respective deadlines. If these packets are not dropped in future scheduling decisions (when new arrivals occur), they can all be served before their deadlines, provided that an *appropriate* service algorithm is selected. The scheduling algorithm guarantees that there are sufficient resources to serve all scheduled packets before they expire. If the work-conserving EDF service algorithm is selected then it is easy to see that all scheduled packets will be served before their deadlines. Other ad-hoc service algorithms may also succeed in serving all scheduled packets.

Since the service algorithm determines the packets which will still be in the system when a future arrival occurs (next decision time) it affects the packet dropping process at that time and, thus, the resulting performance of the proposed SD scheduling policy. A serving algorithm was developed which attempts to serve as many of the scheduled packets as possible using the SJF algorithm without causing any of the remaining packets to violate their deadlines. The rest of the packets are served using the EDF algorithm.

In order to implement this serving algorithm the server uses information that is already available after the scheduling process is completed. It is known which packets have been scheduled and their deadlines. Let t denote the current decision time instant. Let $\mathcal{R}_t = \{P_1, P_2, ..., P_n\}$ denote the set of scheduled packets in the system at time t. Also, let I_i be a counter associated with $\mathcal{T}_i$ whose (initial) value at t is equal to T_i, as it was defined in the scheduling algorithm. Finally, let the total sum of the length of the packets with a common deadline which are associated with a common counter I_i be represented by the variable W_i. The serving algorithm is described next.

The serving algorithm considers the shortest packets first (P_i) and examines whether any of the packets with earlier deadlines (shorter counter values, I_i) will violate them if packet P_i were to be served first followed by the remaining packets served in an EDF fashion (step 3). There is no need to check the packets with later deadlines since it is guarranteed that there are sufficient resources to serve all scheduled packets using the EDF algorithm. Hence, a packet with an earlier expiration date can be served before a packet with a later expiration date without causing any violations. If there is no deadline violation, packet P_i is served and the delay tolerances (counters) are reduced by an amount equal to its length $\hat{P}_i$ (step 3-(a)). If a possible deadline violation is detected, though, packet P_i is not served. Instead, the server will consider those packets whose deadline would have been violated. Again, it will start from the shortest packet and examine whether any violations occur to packets of earlier deadlines (if any). If not, it will proceed to serve those packets fully (step 3-(b)). Finally, the current value of the counters is kept to be used as its initial value when the next packet is considered (step 4).

At the arrival (decision) time t:

1. Create $\mathcal{R}_t$; that is order the packets in the system according to SJF. Create $\mathcal{T}_i$. Initialise counters I_i, $1 \leq i \leq k$.
2. Select the shortest packet say P_i from:
   (a) $\mathcal{R}_t$ if no possible deadline violation has been detected.
   (b) If a possible deadline violation has been detected for the packets belonging to some set say T_i then select the shortest of those packets.

   Suppose that $P_i \in \mathcal{T}_i$ with W_i.
3. Check for $\sum_{l=1}^{m} W_l \leq I_l - \hat{P}_i$ for $1 \leq m < i$:
   (a) If all inequalities hold true set $I_l(0) = I_l$ for $1 \leq l \leq k$; set $W_i(0) = W_i$; set $m = 1$

and repeat for $\hat{P}_i$ steps while packet is being served:

   i. $W_i(m+1) = W_i(m) - 1$.
   ii. $I_l(m+1) = I_l(m) - 1$.

(b) If the inequality does not hold true for the packets belonging to some set then a possible deadline violation has been detected. Go to step 2.

4. Set $I_l = I_l(\hat{P}_i)$; set $W_i = W_i(\hat{P}_i)$; remove P_i from $\mathcal{R}_t$.

The above procedure will seve as many packets as possible according to the SJF approach. If this approach can not be used for all the packets in a specific realisation, the algorithm will resort to serving them in an EDF fashion.

C. *Comments on the optimality of the SD scheduling policy*

For the arrival process considered so far, which assumes bursty traffic delivering application data units at random arrival instants, the scheduler needs to make a new scheduling decision at every new arrival instant aiming to maximise the number of packets served by the expiration of the current deadlines. If there is no interference (i.e. if there are no future arrivals before the expiration of the current deadlines) then all of the scheduled packets will be served and the performance of the proposed policy will be optimal. Since new arrivals may occur before the expiration of the current deadlines, there exist realisations which demonstrate that the *serving* algorithm does not maximise the number of complete packets served and, hence, the resulting performance of the SD policy is suboptimal. In these cases, the proposed algorithm makes an optimal *scheduling* decision aiming to maximises the number of packets that can be served based on the current information about past and present arrivals. However, the *service* procedure may be suboptimal in the sense that it may not serve the maximum number of packets possible between the time slot servicing begins and the time slot new arrivals occur. Such an optimal service procedure may not be possible to implement due to the variable deadlines. This is expected since the proposed policy is not predictive as it does not assume any knowledge about the future bahavour of the traffic. A more elaborate presentation of this section together with a proof on the optimality of the scheduling policy is given in [6].

IV. SPECIAL CASE: SD POLICY WITH STARTING TIME SELECTION

Continuous media traffic can be bursty in nature. When several such data streams are multiplexed the packet loss probability can be high at times where there is peak demand from many sources. In an environment in which the data streams belong to the same type of application (i.e. they have the same period) and are synchronised at the period level this problem becomes more profound.

A simple and straightforward approach would be to shift the starting times of the periodic intervals of all streams with respect to each other by an amount of τ slots. This can be arranged upon connection set up. In order to implement the SD scheduling policy in this environment a new scheduling decision needs to be made at every new arrival instant. The implementation of the original SD scheduling policy can be simplified, however, by imposing some order in the arrival process. This approach is referred to as the SD policy with starting time selection and it is described in more detail in [6].

V. NUMERICAL RESULTS AND DISCUSSION

In this section the performance of the proposed policies is compared to that of the EDF, which serves streams on a packet by packet basis based on their deadlines only, and SJF, which serves streams on a packet by packet basis based on their length only. Comparison is based on numerical results computed through simulations for the packet loss and cell loss probabilities induced by these policies. In this paper only two sets of plots are shown. More results can be found in the CDSP report.

The simulations examples have been chosen to show the realisations under which the difference in performance between the SD policy, the EDF and the SJF is greater. Simulations were run for low as well as high system utilisations. The plots that follow concentrate mostly on high system utilisations. It is under these conditions that the losses are expected to be the highest and the performance of the three policies is examined to assess the difference in the induced losses.

The performance of the SD policy was compared to that of EDF and SJF for two data streams synchronised at the frame level. Two sets of examples were considered. Each set looked at a specific arrival process. The objective was to look at those realisation in which the EDF or the SJF algorithms would cause the majority of the packets to be dropped and compare their performance to this of the proposed policy.

The first set of simulations focused on demonstrating the difference between the performance of the SD policy and the EDF algorithm. There are two data streams with $D^0 < D^1$. Stream 0 has one long packet per period whereas stream 1 has many short packets. The EDF algorithm would always serve the long packet first since it has an earlier expiration date and would result in a large number of small packets from stream 1 to be dropped. The SD policy, on the other hand, is expected to maximise the number of complete packets served as it takes into consideration the size of the packets as well as their deadline and in this case it would give priority to the shorter packets of stream 1. Under these conditions, the SJF algorithm is expected to perform very close to the proposed policy.

Figure 1 shows the packet losses and cell losses for two data streams with period $T = 50$ and deadlines $D^0 = 40$ and $D^1 = 50$. Stream 0 has one packet arrival per period whose length varies between 35-40 cells with a uniform distribution. Stream 1 has many packet arrivals per period, each one of length 1. The system utilisation increases as the number of packets for stream 1 increases following a uniform distribution. The difference in packet losses between the SD algorithm and EDF increases as the system utilisation increases to almost two orders of magnitude whereas the packet loss curve of the SJF follows the proposed algorithm. The cell losses are higher for the SD policy and SJF as compared to those of EDF but they converge as the utilisation of the system approaches 1.4.

The second set of simulations focused on demonstrating the difference between the performance of the SD algorithm and the SJF algorithm. Again, there are two data streams with $D^0 < D^1$. Both streams have many short packets arriving per period but the packets of stream 0 are shorter than the packets of stream 1. In this case, the SJF algorithm would start by serving the packets of stream 1 first, thus, resulting in the loss of the majority of the packets of stream 0 as they have a shorter deadline. The proposed policy, on the other hand, is expected to maximise the number of complete packets served as it takes into consideration the deadlines of the packets as well as their size. Under these conditions, the EDF algorithm is expected to perform very close to the proposed algorithm.

Figure 2 shows the packet losses and cell losses for two streams with period $T = 100$ and deadlines $D^0 = 70$ and $D^1 = 100$. Stream 0 has 15-20 packet arrivals per period, each one of length 2 cells. Stream 1 has variable number of packet arrivals per period each one of length 1. The system utilisation increases as the number of packets for stream 1 increases following a uniform distribution. In this case, the difference in both the packet and the cell losses is more accute for lower utilisations. The SD policy and the EDF shows no losses for utilisations less than approximately 0.95 and lower losses than SJF for a system load between 0.95-1.05. For utilisations above 1.05 the performance curves of all policies tend to converge.

References

[1] Klara Nahrstedt Ian R. Philp and Jane W.S. Liu. Scheduling and Buffer Management for Soft-Real-Time VBR Traffic in Packet-Switched Networks. *Technical Report, University of Illinois at Urbana-Champaign, Ill*, 1996.

[2] Jin-Chyang Jiau Cheng-Shong Wu and Kim-Joan Chen. Characterising Traffic Behavor and Providing end-to-end Service Guarantees Within ATM Networks. *Proceedings of IEEE Infocom*, 1997.

[3] Kwok-Tung LO Jian Feng and Hassan Mehrpour. Simulation Analysis on Statistical Multiplexing og MPEG Video Sources. *Proceedings of IEEE International Conference on Communications*, 1997.

[4] Ioannis Stavrakakis Zoe Antoniou. Earliest Due First Scheduling for Application-Level QoS Delivery. *Proceedings of IEEE Conference on Protocols for Multimedia Systems - Multimedia Networking*, 1997.

[5] Sujata Banerjee. Translating Application Requirements to ATM Cell Level Requirements. *Proceesings of IEEE International Conference on Communications*, 1997.

[6] Z. Antoniou and I. Stavrakakis. Efficient QoS Support of Continuous Media Stream Packets with Variable Size and Diverse Deadlines. *Technical Report, TR-CDSP-98-46, CDSP Centre, Northeastern University*, 1998.

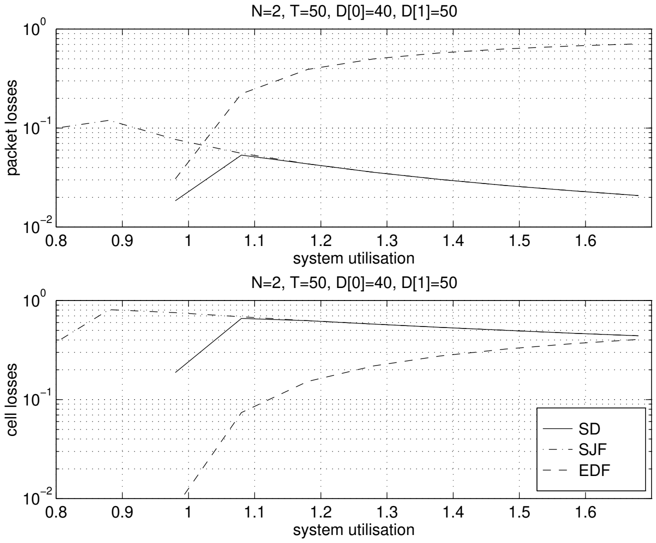

Figure 1: Concentrates on the performance difference between the SD policy and EDF.

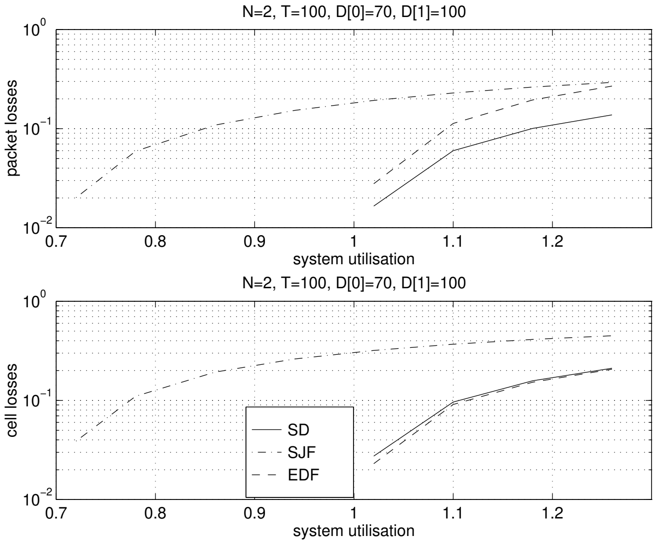

Figure 2: Concentrates on the performance difference between the SD policy and SJF.