Special Issue

Scheduling Algorithms for Application-Level QoS Support of Continuous Media Streams*

ZOE ANTONIOU, IOANNIS STAVRAKAKIS
Department of Electrical and Computer Engineering
Northeastern University
Boston, MA 02115
{*zoe,ioannis*}@*cdsp.neu.edu*

Abstract. The Quality of Service (QoS) of real time applications is affected in a rather direct manner by the timely delivery of Application-level Data Units (ADUs) which are typically mapped into several Network-level Data Units (NDUs). As a consequence, it would be reasonable to associate a common deadline with all the NDUs in one ADU and develop a scheduling policy that aims to maximise the number of ADUs that are delivered by their respective deadlines. In the first part of the paper, an ADU-based Earliest Due First (EDF) scheduling policy is investigated under periodic ADU arrival processes which are typical of continuous media streams. The proposed policy can be implemented in terms of a simple First ADU In - First ADU Out (FAI-FAO) policy and it is shown to improve on the ADU dropping rate induced by FIFO policies of comparable complexity.

In the second part of the paper, the more complex and effective Size-Deadline Sensitive(SDS) scheduling policy is proposed and investigated under general ADU arrival processes. This policy is shown to be optimal in the sense that it maximises the number of complete ADUs scheduled for transmission at any point in time, by taking into consideration the associated diverse deadlines and the ADU sizes and combining the principles of two well known scheduling algorithms, the Earliest Due First (EDF) and the Shortest Job First (SJF). It is argued and illustrated that the proposed optimal scheduling leads in efficient actual service which under certain arrival processes is shown to be the optimal. Numerical results are presented and the induced ADU and NDU loss rates under the proposed policy are compared against those induced by the standard EDF and SJF policies.

1 INTRODUCTION

Future high-speed networks -such as the Next Generation Internet or the Asynchronous Transfer Mode Networks- are expected to support the simultaneous transmission of data, voice and video streams of variable bit rates (VBR) and diverse Quality of Service (QoS) requirements [1], [2]. Real time applications are expected to share a large portion of the traffic transported over such networks. The traffic characteristics of real time applications can be very diverse as they combine a variety of media data such as text, audio, digital video, graphics and images. Such applications have, in general, stringent Quality of Service (QoS) requirements in terms of delay, loss and jitter. The transmission of continuous media traffic is an inherently difficult problem due to the time sensitive nature and the traffic variability of these applications [3], [4], [5].

Consider, for instance, the transmission of video applications in which it is required that frames (images) be transmitted at a certain rate in order to guarantee no starvation or overflow at the receiving end. Roughly every 33.3 ms, the video application reads and packetises a frame, and sends the packets to the network using some protocol stack. Bits generated over a frame form a logical block of information (Application Data Unit or ADU) which is central to both the traffic generation process at the transmitter and the reconstruction process at the receiver. Such a role may be played by a smaller encoding unit than the frame such as a macroblock [6]. Since frames are typically generated at a constant frame rate the nature of such an application is periodic.

ADUs, such as a frame, can be mapped into one or more Network Data Units (NDUs). 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 NDUs of an ADU should be delivered in time for the reconstruction process in order to avoid either starvation or buffer overflow at the receiving end. When network resources are reserved at peak demand, all NDUs of the application data unit will be delivered on

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

time, leading to a timely and error-free reconstruction at the receiver [7], [8], [9].

When resources are over-allocated to a number of applications -to increase network utilisation- all NDUs of an ADU may not be delivered before this ADU is required by the reconstruction process. In this case, ADUs may be significantly distorted due to missing NDUs or be impossible to process at all. The required QoS for such applications may then be defined in terms of a metric associated with the correctly and timely delivered ADUs. An application-level QoS metric such as the ADU loss rate would describe more accurately the QoS delivered to the end user, than a network-level QoS metric such as the NDU 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 ADU should be received before it can be processed by the receiver, then it would be reasonable to associate a common deadline with all the NDUs in the same data unit and develop scheduling policies that aim to maximise the number of complete ADUs that are delivered by their respective due date. This approach is different from many traditional approaches in which every NDU is assigned its own deadline and the scheduling policy typically attempts to maximise the number of individual NDUs transmitted by their due date [10], [11].

This paper investigates two major scheduling approaches supporting applications which require application-level QoS. The approach discussed first is referred to as the First ADU In - First ADU Out (FAI-FAO) policy. Unlike the typical NDU-based Earliest Due First (EDF) scheduling policy which requires sorting on a NDU-by-NDU basis [12], the proposed policy requires such sorting on an ADU-by-ADU basis, reducing the processing burden. For continuous media applications in which ADUs are delivered over well defined periods the processing burden of the proposed policy can be substantially reduced by ordering the multiplexed data streams according to their respective deadlines and serving them according to the FAI-FAO policy, as it will become clear in later sections. The complexity of this policy is similar to that of a (NDU-based) FIFO policy applied to the deadline ordered streams. The performance of the FAI-FAO and the FIFO policies is evaluated in terms of the induced NDU loss and ADU loss probabilities. In view of the earlier discussion the ADU loss probability may be a more relevant metric for applications in which higher layer protocol data units are fragmented into a number of NDUs; these ADUs may be considered to be lost if any one of their NDUs are lost, unless some redundancy is introduced in the data unit to improve the probability of correct reception at the expense of increasing the traffic in the network.

As demonstrated by the results, the FAI-FAO policy does improve on the ADU loss probability with respect to the NDU-based policies it is compared to. However, there exist realisations in which it fails to maximise the number of complete ADUs served. This is due to the fact that the scheduling decisions are taken based on the ADU deadlines alone without considering their size. Also, past scheduling decisions regarding ADUs which are still in the system are not reconsidered when new arrivals to the system occur. Such issues are taken into consideration in the design of the second scheduling policy, the Size-Deadline Sensitive (SDS) based policy which is briefly introduced next.

The SDS scheduling policy aims to maximise the number of ADUs *served* by their deadlines from all data streams by maximising the number of ADUs *scheduled* (as opposed to actually *served*) 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 ADUs as well as their relative deadlines combining principles from the Earliest Due First (EDF) and the Shortest Job First (SJF) algorithms which are shown to be suboptimal for the environment considered here. It is argued and illustrated that the proposed optimal scheduling leads to efficient actual service which under certain arrival processes is shown to be the optimal.

The scheduling policies investigated in this paper are considered in an environment where N application streams share one transmission resource. The system time is slotted with one slot being equal to the time required for the transmission of one NDU. A variable with superscript j denotes a quantity associated with stream j, $0 \leq j \leq N-1$. Only complete ADUs are scheduled for service; an ADU is considered to be lost if at least one of its NDUs is lost.

2 Study of the FAI-FAO scheduling policy

2.1 Description of the policy

Each one of the N application streams is assumed to deliver to the scheduler periodic ADU traffic with a common period T. Let A^j denote the generic random variable describing the length (in NDUs) of the ADU delivered by stream j at a random periodic instant; let $\hat{A}^j$ denote its mean. It is assumed that an ADU of size A^j delivers to the scheduler the A^j associated NDUs instantaneously. The study is also applicable if the NDUs are delivered over A^j consecutive time slots, starting from the associated periodic instant; this would be the case if the N streams are delivered to the scheduler over N incoming lines of the same speed as the outgoing one. In such a case, it would be necessary that the ADU size is fixed and known, or be provided by the first NDU of the ADU. In addition, it is assumed that information regarding the length of an ADU be available to the scheduler upon ADU arrival. Let D^j denote the Maximum Delay Tolerance (MDT) -in slots- of an ADU from stream j; D^j represents the maximum time which the ADU may remain in the system following the periodic instant marking its arrival.

The per stream periodic (potential) arrival instants form a sequence of frames of length T at the boundaries of which an ADU arrival is possible only. Let $\{t_k\}_{k\geq 0}$ denote the sequence of ADU arrival instants (from any stream); t_k indicates the arrival instance of the k^{th} ADU. Variables with subscript k will represent the value of the associated quantity at time t_k. Let J_k denote the stream for which t_k is an ADU arrival instant. Let τ_k denote the interval between two consecutive ADU arrivals (from any stream); that is $\tau_k = t_{k+1} - t_k$ with $\tau_k = \tau^j \quad \text{if} \quad J_k = j$, where τ^j denotes the interarrival time between two consecutive ADUs from streams j and $(j+1)mod(N)$ which represents the amount of shifting between consecutive frame boundaries and it is equal to zero if these streams are synchronised. In the rest of this paper it is assumed that the streams are ordered in terms of increasing deadlines. That is, $D^j \leq D^{j+1}$ for $0 \leq j \leq N-2$.

As it was stated earlier, the goal of the proposed policy is the timely delivery of complete ADUs. This goal suggests that the scheduler should serve ADUs based on the Earliest Due First (EDF) policy. In view of the deadline ordering and the periodicity of the (per stream) ADU arrival process it is easy to establish that a scheduler that serves ADUs according to the *First In First Out* policy does serve them according to the EDF policy. This proposed policy is referred to as the First ADU In - First ADU Out (FAI-FAO) policy and is discussed in more detail below.

Upon ADU arrival the scheduler decides whether this ADU can be served by its deadline or not. In the former case, the ADU is served before its deadline expires as described below. In the latter case, the *entire* ADU is discarded. Both the description of the policy as well as its implementation is facilitated by introducing the *service queue* to which NDUs are shifted for storage before they are transmitted. All NDUs shifted to the service queue will be transmitted, following the first NDU in - first NDU out policy.[1]

Suppose that an ADU from stream j of size A^j arrives at time t_k when the service queue occupancy is equal to Q. If:

$$Q + A^j \leq D^j \tag{1}$$

the ADU is shifted to the service queue. This condition guarantees that the NDUs will be sent to the output link within D^j time slots and, thus, they will meet their deadline. If the associated logical queue overflows, then the dropped NDUs will be the ones to miss the ADU deadline. In this case no NDU of the ADU is shifted to the service queue (this is relaxed in partial FAI-FAO policy discussed later). Notice that the FAI-FAO scheduling policy is *work conserving* in the sense that the server will not remain idle as long as there are ADUs waiting to be served.

A variation of the FAI-FAO policy -referred to as the partial FAI-FAO (p-FAI-FAO) policy- is also investigated here. This policy shifts to the service queue the portion of an ADU which can meet its deadline; that is, ADUs are served in part as well and, consequently, the number of complete ADUs served can be only smaller than that under the FAI-FAO. The partial FAI-FAO policy would be meaningful only if partial ADUs are still of use to the user or the length of the ADU is not available to the scheduler (to carry out the test in (1)). Both the ADU and NDU dropping rates induced by a FAI-FAO or a partial FAI-FAO are calculated analytically. The derivation may be found in Appendix A. Some numerical results are presented in the next subsection.

2.2 NUMERICAL RESULTS AND DISCUSSION

In this section, the performance of the FAI-FAO policies is compared to that of the FIFO policy which serves the deadline ordered streams on a NDU-by-NDU basis.

The induced losses under the proposed policies have been computed using the derivation presented in Appendix A while those under the FIFO policy have been computed through simulations. Two versions of the FIFO policy have been considered: FIFO policies with and without an expired NDU discarding mechanism. The first one (pure-FIFO or p-FIFO) assumes no knowledge of the deadlines and it is simple and straightforward in its implementation. The FAI-FAO policy is compared to the p-FIFO in order to evaluate their relative performances and to determine if a significant gain can be achieved by employing the slightly more involved FAI-FAO policy. The second FIFO policy that is considered (d-FIFO) has an expired NDU discarding mechanism. It identifies and drops those NDUs that can not meet their deadline. In order for the expired NDUs to be dropped they typically need to be searched for. The computational overhead in this case is quite significant for a large number of streams. Another drawback of the FIFO approach is that NDUs from different ADUs are multiplexed at the service queue. This introduces distortion to the original NDU stream and NDUs are spread over longer time intervals.

Initially a homogeneous traffic environment is considered whereby the traffic from all data streams is statistically identical and have a common maximum delay tolerance). The ADU lengths are geometrically distributed. The interarrival interval τ_k is constant and equal to $\tau = \frac{T}{N}$ (τ is always rounded to the nearest integer). The system utilisation can be varied in proportion to the average ADU length and the period T.

Fig. 7 shows the ADU loss probability under the FAI-FAO policy, the p-FAI-FAO policy and the FIFO as a function of the system utilisation for $N = 8$, $T = 60$ and identical deadlines $D = 60$ (1 frame). Among the four policies, FAI-FAO provides the lowest ADU loss rate. For

[1] In this section, it is assumed that each stream delivers at most one ADU at any arrival instant. However, the algorithm can be applied when multiple ADU arrivals occur. In this case, the scheduler could shift the ADUs to the service queue either in a random order or in order of increasing size. The latter would ensure that a larger number of ADUs would be served than under the random ADU shifting.

low system utilisation it performs better than the FIFO approaches by approximately an order of magnitude. As the system utilisation increases the p-FAI-FAO and the d-FIFO and the FIFO tend to converge to a common ADU loss rate whereas the losses of the p-FIFO increase at a faster rate.

Fig. 8 shows the NDU loss rates for the cases considered in Fig. 7. Notice that p-FAI-FAO outperforms the FAI-FAO as expected. However, the performance of FAI-FAO is the same or better than that of the p-FIFO.

From the above results it can be clearly concluded that FAI-FAO has the lowest ADU loss rate while at the same time it keeps the NDU loss rate at the same or lower level than that of the p-FIFO approach. The performances of p-FAI-FAO and of the d-FIFO are generally in the middle with higher ADU loss rates and slightly lower or equal NDU loss rates than FAI-FAO.

Similar behavior regarding the ADU and NDU losses has been observed in the results derived for $N = 8$, $T = 64$, $D = 96$ ($1\frac{1}{2}$ frame) and for $N = 4$, $T = 40$, $D = 40, 60$. Increasing the deadline of the input traffic to 2 frames or more (not shown in this paper) resulted in about the same performance under all policies. In other words, if the deadlines are long enough the losses become independent of the particular scheduling policy and they depend on the traffic load alone.

Fig. 9 and 10 present the the induced ADU and NDU losses, respectively, under all four policies under an ADU size distribution delivering an ADU of length L NDUs with probability p and length 1 with probability $1 - p$ ($1 - L$ bursty periodic traffic model). The system parameters are: $N = 8$, $T = 64$, $L = 36$, $p = 0.2$ and deadlines equal to 1 frames (also, for $1\frac{1}{2}$ frames not shown here). The same behavior is observed as under the geometric ADU arrivals.

An inhomogeneous traffic environment has also been considered. The objective in this case is to evaluate the level of diversity in the induced performance. Out of the $N = 8$ data streams, two are assumed to deliver probabilistically larger ADUs and since the period is the same for all streams, have larger load. τ is maintained constant for all streams. The six (background) streams are assumed to deliver ADUs of size 1 or L with parameter p (as before) with $L = 10$ and $p = 0.2$. The two (more active streams) streams have parameters L and $p = 0.4$, where L assumes a value between 8 and 56 so that a specific system utilisation is achieved.

Fig. 11 shows the ADU loss rates for the two more active streams (data streams 1 and 4) and the six background streams under all four policies. It can be observed that FAI-FAO induces the lowest losses for the two more active streams which are of the order of $10^{-2} - 10^{-3}$ and are observed only for a system load higher than 0.7. The other policies induce higher losses for a wider range of system utilisation. The two FIFO policies induce approximately the same losses which are higher than those under FAI-FAO by at least an order of magnitude. For the six background streams FAI-FAO induces a loss rate of the order of $10^{-3} - 10^{-4}$ for system utilisation higher than 0.7 only.

To further reduce the ADU losses an attempt was made to adjust the streams' interarrival time τ in proportion to the *expected ADU length* of each stream. Increasing the values of τ for streams with high mean ADU length could result in a reduction of the average occupancy of the service buffer since "busy" streams will now have more time to be served before the next ADU arrival occurs. The value of τ has been set as: $\tau^j = (\overline{A_T^j})\mathrm{T}/(\sum_{j=0}^{N-1} \overline{A_T^j})$ and is rounded to the nearest integer as before.

Fig. 12 shows the ADU losses for FAI-FAO with equal values of τ (solid line) and with weighted values of τ (dashed line). It can be noted that adjusting the value of τ hardly affects the loss rate for the two more active streams. In fact, there is a slight increase. On the other hand, the loss rate for the six background has been improved significantly. Losses are below the order of 10^{-5} only for system utilisations below than 0.95.

The more active streams have longer ADUs as compared to other streams. Long ADUs will reserve more space in the service buffer once they arrive and are accepted by the system. Suppose that a more active stream is followed by a few less active ones which is followed in turn by a more active one and so on. This is the case for the ADU losses in Fig. 12. Once a long ADU is accepted by the system it will be shifted to the service buffer. The longer the interarrival interval τ is the more NDUs will be served before the next ADU arrives thus creating more free slots in the service buffer for the next arrivals. This implies that fewer ADUs will be dropped from the less active streams which follow (decrease in the loss rate for the less active streams as seen in Fig. 12). On the other hand, if the interarrival interval τ is the same for all streams then shorter ADUs from less busy streams may have to be dropped more frequently.

3 Study of the SDS scheduling policy

3.1 Introduction

Although the FAI-FAO policy improves on the ADU loss performance induced by NDU-based policies such as the FIFO, it is not optimal in the sense that it does not minimise the ADU loss rate. This is due to the fact that the FAI-FAO policy schedules an ADU upon arrival based on its deadline only and never revises its decision. As a consequence, earlier scheduled long ADUs may result in dropping a larger number of later arriving ADUs; thus, the ADU dropping rate is not minimised. In this section a policy which takes into consideration the ADU size in addition to the deadline and revisits past scheduling decisions in the future is proposed. This policy is referred to as the Size-Deadline-Sensitive (SDS) policy and maximises the number of ADUs scheduled based on the known workload.

Some related existing policies are reviewed below in an attempt to illustrate their inefficiencies and motivate the proposed policy.

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. This policy will maximise the number of (fixed size) NDUs served by their respective deadlines but would fail to maximise the number of ADUs served by their respective deadline. This is illustrated in the following example.

Consider the application of the EDF policy for scheduling five ADUs coming from two applications (streams) with diverse deadlines; let $D^1 < D^2$. The ADUs are numbered from 1 to 5 in order of increasing size: ADUs 1, 2 and 5 (3 and 4) have deadline D^1 (D^2). Let t denote the time when all arrivals occur (simultaneously) or the time at which the scheduling decision is taken.

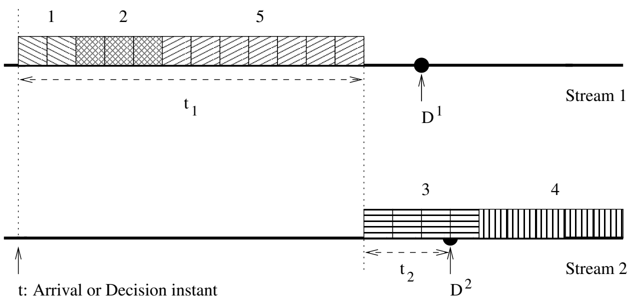

Figure 1: *Example showing the suboptimality of the EDF policy.*

As illustrated in Fig. 1, the EDF policy would serve ADUs 1, 2 and 5 first (for a total service time of t_1 slots), serve ADU 3 in part (for a service time t_2 slots) and drop ADU 4. Thus, it would serve 3 out 5 ADUs completely. It is easy to establish that the EDF does not maximise the number of ADUs served in this realisation. For instance, a scheduling policy which takes into consideration both the size and the respective deadlines would serve ADUs 1, 2, 3 and 4 (in this order) and drop ADU 5. Thus, it would serve 4 out of 5 ADUs completely.

The Shortest Job (ADU) First (SJF) policy is known to maximise the number of ADUs served over any time interval, when ADUs do not expire (no deadlines). When ADUs do have deadlines the SJF policy will not necessarily maximise the number of (complete) ADUs served, as the following example demonstrates.

Consider the application of the EDF policy for scheduling ADUs in a system similar to that discussed in conjunction with Fig. 1. Under the ADU realisation shown in Fig. 2, the SJF policy would serve ADUs 1 and 2 (for a total service time t_1 slots), followed by ADUs 3 and 4 (for a total service time t_2 slots and would drop ADU 5 as it can be served only in part. If the diverse deadlines of the ADUs had been taken into consideration, though, together with their size, the scheduler could serve ADUs 1, 2 and 5 followed by ADUs 3 and 4. So there would be no losses.

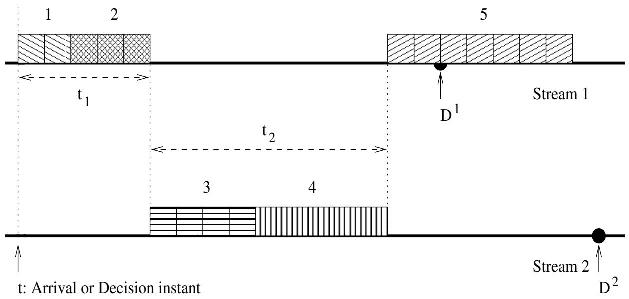

Figure 2: *Example showing the suboptimality of the SJF policy.*

The EDF and SJF policies described above either focus on the deadline of an ADU or its size in order to make scheduling decisions. The examples have clearly demonstrated that for the arrival processes considered here (variable ADU lengths and deadlines) there exist realisations for which these policies fail to maximise the number of complete ADUs served by their respective deadlines. The policy proposed in the next section schedules and serves ADUs using the basic principles of both the EDF and the SJF policies. It makes decisions based on both the size of the ADUs and their relative deadlines.

3.2 Description of the SDS policy

The proposed SDS policy aims to maximise the number of unexpired complete ADUs served over any time interval by taking into consideration both the deadlines and the sizes of the ADUs. Although the SDS policy may not maximise the number of unexpired ADUs 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 environment considered here may be viewed as consisting of one arrival process delivering ADUs belonging in different classes (with different deadlines) or, equivalently, as consisting of multiple streams each with its own deadline. The latter model is adopted in this paper. The per stream arrival process is assumed to deliver (possibly multiple) ADUs at random times (not necessarily periodic). Each ADU has variable size (in NDUs) and a maximum delay tolerance D. The SDS policy is executed upon ADU arrival. At this time, all ADUs present in the system are scheduled for transmission. New scheduling decisions are taken for ADUs which arrived before the current ADU arrival time instant and they are not fully transmitted.

As it will become clear below, the SDS 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).

3.2.1 The scheduling algorithm

The scheduling algorithm is the central component of the proposed SDS scheduling policy and determines which ADUs will be scheduled for service and which ones will be dropped at the current decision time without any further consideration. The ADUs scheduled (for service) will actually be served completely and before the expiration of their respective deadlines provided that no future arrivals interfere. If some of the the scheduled ADUs are in the system in the next decision (ADU arrival) instant, some of these ADUs may actually be dropped at that time, since all ADUs 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 ADUs from other streams which can be scheduled. It is easy to see that the impact is minimised if this ADU 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 ADUs may expire earlier. Notice that the actual service algorithm (and not the scheduling algorithm) will determine the actual service order.

As an example consider the problem of scheduling at time t an ADU from stream 2 in the two stream system shown in Fig. 3, where $D^1 < D^2$. ADUs of stream 1 (2) can be scheduled over the interval $[1, D^1]$ ($[1, D^2]$). If stream 2 uses the first D^1 slots, it will reduce the number of slots available for service for stream 1. If it uses the last $D^2 - D^1$ slots it will have no affect on stream 1 as its deadline will have expired after the first D^1 slots and the effect on the number of slots for stream 2 is the same. Thus, when scheduling ADUs from streams with variable deadlines, a stream with a latter deadline could first utilise the slots after the deadline expiration of streams with earlier deadlines. In this way, the available time over which service can be provided to the other streams is maximised.

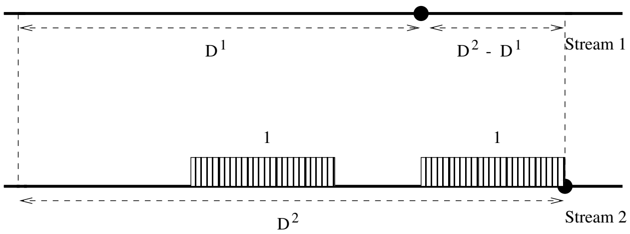

Figure 3: Example illustrating the SDS scheduling principle.

Let t denote the current decision (arrival) time instant. Let $\mathcal{P}_t = \{P_1, P_2, ..., P_n\}$ denote the set of ADUs in the system at time t, ordered according to increasing size or remaining service time; let $\hat{P}$ denote the size of P. Let $\mathcal{S}_t = \{T_1, T_2, ..., T_k\}$ denote the set of the (remaining) delay tolerances of the ADUs 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 $\mathcal{T}_i = \{P$: delay tolerance of P 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 ADU first (P) and examines whether it can be served within its delay tolerance (step 3-(a) below). If not, it considers the next shortest ADU. Otherwise, over the next $\hat{P}$ iterations it reduces the delay tolerances (counters) whose value in the current iteration is higher than or equal to the current delay tolerance of this ADU (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 ADU is considered (step 4). The following steps describe the operations of the algorithm.

At the arrival (decision) time t:

1. Create $\mathcal{P}_t$. Create $\mathcal{T}_i$. Initialise counters I_i, $1 \leq i \leq k$. $I_l(0) = I_l$, $1 \leq l \leq k$.

2. Select the shortest ADU in $\mathcal{P}_t$, say P, and remove it from $\mathcal{P}_t$.

3. Suppose that $P \in \mathcal{T}_i$ then:

   (a) If $I_i - \hat{P} < 0$, drop P and repeat step 2.

   (b) Else, set $I_l(0) = I_l$ for $1 \leq l \leq k$; set $m = 1$ and repeat for $\hat{P}$ 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 $I_l = I_l(\hat{P})$ and go to step 2.

Notice that if the counter associated with the currently considered ADU, 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 ADU is scheduled (so far) over slots beyond the deadline associated with counter I and, thus, no reduction in the delay tolerance of such ADUs 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 ADUs with the delay tolerance indicated by I.

The following two examples illustrate the operation of the proposed scheduling algorithm. Fig. 4 shows two data streams with deadlines $D^1 < D^2$. There are two ADUs to be scheduled; $\mathcal{P}_t = \{P_1, P_2\}$. The smallest ADU P_1 is associated with stream 2 and has deadline D^2 (associated counter I_2); ADU P_2 is associated with stream 1 and has deadline D^1 (associated counter I_1). The initial values of the counters are D^2 and D^1 respectively. ADU P_1 is scheduled first by following the steps (2) and (3) of the scheduling algorithm and decrementing counter I_2 over the first $D^2 - D^1$ iterations (step (3)-(b)) and both counters I_2 and I_1 over the following $\hat{P}_1 - (D^2 - D^1)$ iterations. ADU P_2 is scheduled next. The current values of the counters are: $I_1 = D^1 - [\hat{P}_1 - (D^2 - D^1)]$, $I_2 = D^2 - \hat{P}_1$. Both counters are decremented over the next $\hat{P}_2$ iterations. The current scheduling is completed after $\hat{P}_2$ iterations. The iteration steps are mapped into slots in Fig. 4 for illustrations purposes only since our iteration step corresponds to one time slot in actual delay.

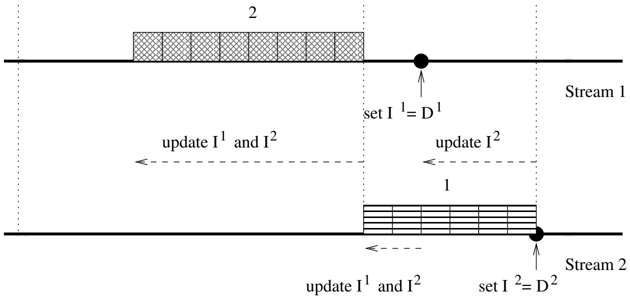

Figure 4: *When I_2 drops below the counter of stream 1, I_1 is updated to the value of I_2. Likewise, I_2 is decreased along with I_1 when scheduling ADU P_2.*

The second example is shown in Fig. 5. There are 2 streams with deadlines $D^1 < D^2$. There are two ADUs to be scheduled $\mathcal{P}_t = \{P_1, P_2\}$. The smallest ADU P_1 is associated with stream 1 and has deadline D^1 (associated counter I_1); ADU P_2 is associated with stream 2 and has deadline D^2 (associated counter I_2). The initial values of the counters are D^1 and D^2 respectively. ADU P_1 is scheduled first by following the steps (2) and (3) of the scheduling algorithm and decrementing both counters I_1 and I_2 for $\hat{P}_1$ iterations. ADU P_2 is scheduled next. The current values of the counters are: $I_1 = D^1 - \hat{P}_1$ and $I_2 = D^2 - \hat{P}_1$. Only counter I_2 is decremented over the next $\hat{P}_2$ iterations. The current scheduling is completed after $\hat{P}_2$ iterations.

An ADU P which is currently being served may be interrupted by the arrival of new ADUs. In this case, the unserved portion of P can be treated as an ADU itself and be considered for a new scheduling decision together with the rest of the unserved ADUs and the new arrivals. If the remaining ADU is not dropped in the current scheduling cycle it may be the first to be served or be served later as the service algorithm would specify. Another option would be not to interrupt the service of P. This would result in sub-optimal scheduling if the unserved part of P is not smaller than the size of some of the new ADU arrivals.

It is evident from the above description that the SDS scheduling policy gives priority to the shorter ADUs. This is a necessary condition in order to meet the objective of the

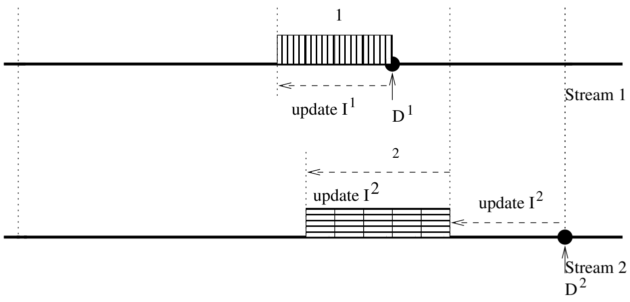

Figure 5: *The counter I_2, is decreased along with the counter I_1 when ADU P_1 is scheduled. Then ADU P_2 is scheduled using the updated contents of I_2.*

scheduling algorithm which is to schedule the maximum number of complete ADUs.

3.2.2 *The service algorithm*

The scheduling algorithm described above identifies the maximum number of ADUs which can be served before their respective deadlines. If these ADUs 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 ADUs before they expire. If the work-conserving EDF service algorithm is selected then it is easy to see that all scheduled ADUs will be served before their deadlines. Other ad-hoc service algorithms may also succeed in serving all scheduled ADUs.

As it will become clear in the next subsection, the EDF algorithm is not the most effective under certain ADU arrivals. For this reason, an alternative more effective service algorithm has been developed and described in Appendix B. This algorithm is referred to as the combined SJF/EDF service algorithm and attempts to serve as many of the scheduled ADUs as possible using the SJF algorithm without causing any of the remaining ADUs to violate their deadlines. The remaining of the ADUs are served using the EDF algorithm.

The following section clarifies the properties of the EDF and the combined SJF/EDF service algorithms, in the environment considered in this paper, in conjunction with the properties of the SDS scheduling algorithm.

3.2.3 *Comments on the effectiveness of the SDS scheduling policy*

The effectiveness of the SDS policy is determined by the effectiveness of the associated scheduling and service algorithms. The following Lemma (proved in Appendix C) establishes the effectiveness of the scheduling algorithm.

Lemma 1 *The scheduling algorithm of the SDS policy is optimal in the sense that for a given set of ADUs present it* *minimises the number of ADUs dropped by this algorithm.*

The above lemma implies that the scheduling algorithm schedules the maximum number of ADUs that can be scheduled for a given set of ADUs present at the decision instant as compared to any other algorithm. Thus, the effectiveness of the SDS policy may be compromised only by an ineffective service algorithm. The following qualities may be attributed to an effective service algorithm:

Definition 1 *A service algorithm is consistent iff it serves all ADUs not dropped by the scheduling algorithm provided no future arrivals interfere.*

Definition 2 *A service algorithm is causally optimal iff it is consistent and it maximises the number of ADUs served since the most recent decision time and for a given set present.*

Definition 3 *A service algorithm is optimal iff -in conjunction with the optimal scheduling algorithm- minimises the overall ADU dropping rate.*

Definition 4 *The remaining arrival process is defined to be the process of the (non-dropped) past arrivals found in the system when new arrivals occur.*

Under the general ADU arrival process considered in section 3 it is easy to establish that both the EDF and the combined SJF/EDF service algorithms are consistent. Similarly, it can be seen that the combined SJF/EDF service algorithm is causally optimal -unlike the EDF service algorithm- as the SJF function guarantees the maximisation of the number of ADUs served since the last decision instant, while the EDF part maintains consistency. The SJF/EDF service algorithm minimises the remaining arrival process by minimising the work backlog present at the future decision time which competes for resources with the new arrivals.

Finally, none of the two service algorithm is optimal since if the remaining arrival process is non-zero, these remaining arrivals may not be the optimal set to compete for resources with a specific set of new arrivals. If the future arrival process is known it is possible to employ ad-hoc service algorithms which will induce a remaining arrival process which can better compete with the known new arrivals. Based on the above the following lemma is evident:

Lemma 2 *Both the EDF and the combined SJF/EDF service algorithms are optimal provided that the remaining arrival process is zero.*

Based on Lemma 1 and 2 the following theorem is evident:

Theorem 1 *Under any ADU arrival process which results in zero* remaining arrival process *the SDS policy is optimal in the sense that it minimises the ADU loss rate.*

Corollary 1 *Under a periodic ADU arrival process the SDS policy is optimal provided that the period is greater than the maximum delay tolerance.*

3.2.4 *Illustration of the SDS scheduling policy*

The following example, (see Fig. 6), illustrates the proposed policy as it is described above. For simplicity only two data streams are considered in the example with deadlines $D^1 < D^2$. Stream 1 has two ADUs and stream 2 has three ADUs. The ADUs are ordered from 1 to 5 according to their size.

Fig. 6(a) shows the ADUs that need to be served from both streams and their relative deadlines. There are two distinct deadlines so two counters are needed. Counter I_1 is set to $D^1 = 16$ slots and counter I_2 is set to $D^2 = 20$ slots.

Fig. 6(b) shows how the scheduling decisions are made. ADU 1 (the shortest) is scheduled first starting from D^1. Counter I_1 decreases to $16-2 = 14$ slots. At the same time counter I_2 is also decreased by 2 slots to $I_2 = 20-2 = 18$, since $I_2 > I_1$. Then ADU 2 is scheduled starting from the current content of I_2 which is decreased to $18 - 3 = 15$ slots. Counter I_1 is not updated during these iterations. Next, ADU 3 is scheduled and counter I_2 is decreased to $15 - 4 = 11$. At this point counter I_1 is also updated to the value of I_2, $I_2 = I_1 = 11$ as counter I_2 becomes less than I_1. For subsequent ADU scheduling both counters are updated at the same time since the remaining slots can be useful to either stream. ADU 4 is scheduled next and $I_1 = I_2 = 11 - 5 = 6$ slots. ADU 5 is dropped as it can not be served fully before its deadline. So the scheduler decided that it can only serve 4 out of 5 complete ADUs.

Once the scheduling decision has been made, the ADUs can be served by employing the EDF service algorithm resulting in the order shown in Fig. 6(c). ADUs 1 and 4 are served first (in no particular order) followed by ADUs 2 and 3 (again in no particular order). Alternatively, the combined SJF/EDF service policy could be employed resulting in serving the ADUs in the following order: 1, 2, 3 and 4 as shown in Fig. 6(d). As stated earlier, both service policies succeed in actually serving all scheduled ADUs. Notice though that the number of ADUs served by any given time instant t following the scheduling time instant under the combined SJF/EDF service policy can only be the same as or larger than the number served under the EDF policy. For example, this is true for time instant t_1 in Fig. 6(c) and (d) where two complete ADUs have been served under the SJF/EDF approach as compared to one and a half under the EDF algorithm.

3.3 The SDS policy under multiple, non-synchronised periodic ADU streams

In this section a simplified implementation of the SDS scheduler is presented under multiple, non-synchronised, periodic ADU streams with a common period (T) and potentially different maximum delay tolerances (D^j), delivering ADUs (instantaneously) at the periodic instants; without loss of generality, it is assumed that the periods of the

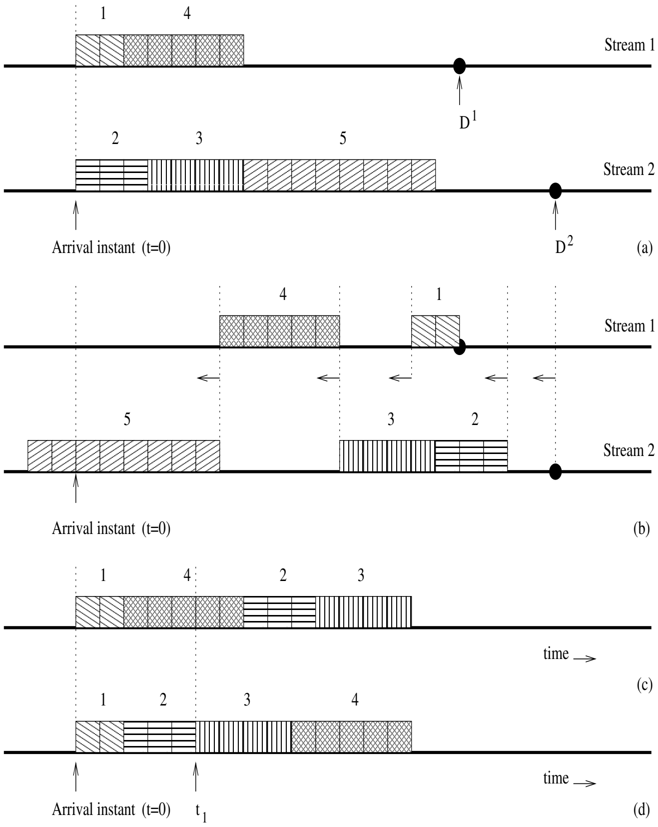

Figure 6: *An example illustrating the SDS scheduling policy; part (a) shows the ADUs that need to be served and their deadlines; part (b) shows the scheduling procedure* backwards *from the deadlines starting with the shortest ADU; parts (c) and (d) show the service order of the EDF and SJF/EDF algorithms respectively.*

streams are shifted equally by τ slots to achieve a uniform de-synchronisation. This system can be employed in modeling, for instance, a video server equipped with an SDS scheduler.

The traffic streams associated with video applications do exhibit periodicities due to the underlying picture/frame representation, as well as variability due to the employed information redundancy reducing encoding schemes. In fact, encoding may be applied to a smaller portion of a frame such as a slice or a macroblock which also exhibit periodicities. When all video application are generated by the same encoding mechanism it is reasonable to assume that the periods are identical. The maximum delay tolerance may be the same for all streams or different. The latter may be the case if applications have quite diverse routes to the destination (in length), or the destination end users have quite diverse buffering (for jitter absorption) capabilities. Finally, the periods of the streams could be de-synchronised due to a designed (uniform) randomisation aiming to reduce the buffer requirement at the server, or a network node receiving multiple video streams. For instance, if the ADUs from all streams are delivered to the server (node) at the beginning of the synchronised periods, most of them will need to be stored and processed later. De-synchronising the periods would result in a smooth traffic presented to the server (node). Period de-synchronising ideas have been explored extensively in the context of multiplexing MPEG video streams at the frame or group of pictures level [13], [14]. By shifting periods the periodically generated, network resource demanding, I frames from different video streams are spread, reducing the peak rate of the aggregated traffic delivered to the server (node).

Under the ADU streams described above the SDS scheduling algorithm is significantly simplified as discussed below. The simplified implementation requires that streams be ordered according to increasing deadlines. That is, the deadline of an ADU expires before that of a later arriving ADU. This condition is met always in the case in which all applications have the same maximum delay tolerance and it requires proper streams ordering to hold otherwise.

Both the description and the implementation of the SDS policy in this environment is facilitated by employing a service queue; its content is denoted by Q. The following steps are executed at decision (arrival) instants:

1. The *new* ADUs from stream j are ordered according to the SJF order.

2. The shortest ADU, say P, of size $\hat{P}$, is selected.

   (a) If $Q + \hat{P} \leq D^j$, P will meet its deadline and it is shifted to the end of the queue.

   (b) If $Q + \hat{P} > D^j$ then the longest ADU in the service queue is compared to $\hat{P}$. If greater than P, then it is removed from the service queue and P is shifted to the end of the service queue; otherwise, P is dropped.

3. P is removed from the sorted list of new arrivals. Step 2 is repeated until the sorted list is empty.

Notice that if step 2.a is simplified to simply rejecting P without looking for the longest previously scheduled ADU in the system, the resulting policy becomes the FAI-FAO policy discussed in section 2 (no reconsideration of past decisions). If ADUs are serviced by applying the FIFO service policy to the service queue, then ADUs are in effect served under the EDF service algorithm. Additional processing is required in order to employ the combined SJF/EDF service policy (Appendix B), if it is desirable.

3.4 Numerical results and discussion

In this section the performance of SDS is compared to that of the EDF and SJF policies. Comparison is based on numerical results obtained through simulations for the ADU loss and NDU loss probabilities induced by these policies.

The simulations examples have been chosen to generate the realisations under which the difference in performance between the considered policies is significant. The results presented concentrate mostly on high system utilisations. Network resources are often over-allocated to increase the statistical multiplexing gain of the network. As a result, the demand for bandwidth will sometimes exceed the available capacity. It is under these conditions, when the system is over-utilised, that the losses are expected to be the highest and the performance of the all four policies is examined to assess the difference in the induced losses.

The first set of simulations demonstrates the difference between the performance of the SDS and the EDF scheduling policies by considering two periodic, synchronised ADU streams with common period T and $D^0 < D^1$. Stream 0 delivers one long ADU per period whereas stream 1 delivers many short ADUs. The EDF policy would always serve the long ADU first since it has an earlier expiration date and would result in a large number of small ADUs from stream 1 to be dropped. The SDS policy, on the other hand, is expected to maximise the number of complete ADUs served as it takes into consideration the size of the ADUs as well as their deadline and in this case it would give priority to the shorter ADUs of stream 1. Under these conditions, the SJF algorithm is expected to perform very close to the proposed policy.

Fig. 13 shows the ADU and NDU losses for two synchronised period ADU data streams with period $T = 50$ and deadlines $D^0 = 40$ and $D^1 = 50$. Stream 0 delivers one ADU per period whose length varies between 35-40 NDUs with a uniform distribution. Stream 1 delivers many ADUs per period, each one of length 1. The system utilisation increases as the number of ADUs for stream 1 increases in steps of five following a uniform distribution. The difference in ADU losses between the SDS and EDF policies increases as the system utilisation increases to almost two orders of magnitude whereas the ADU loss curve of the SJF policy follows that of the SDS policy. Notice that the NDU losses are higher under the SDS and the SJF policies as compared to those of EDF.

The second set of simulations demonstrates the difference between the performance of the SDS and the SJF policies by considering two periodic, synchronised, ADU streams with common period T and $D^0 < D^1$. Both streams deliver many short ADUs per period but the ADUs of stream 0 are shorter than the ADUs of stream 1. In this case, the SJF policy starts by serving the ADUs of stream 1 first, thus, resulting in the loss of the majority of the ADUs of stream 0 as they have a shorter deadline. The SDS policy, on the other hand, maximises the number of complete ADUs served as it takes into consideration the deadlines of the ADUs as well as their size. Under these conditions, the performance of the EDF policy follows that of the SDS policy.

Fig. 14 shows the ADU and NDU losses for two synchronised, periodic ADU streams with period $T = 100$ and deadlines $D^0 = 70$ and $D^1 = 100$. Stream 0 delivers 15-20 ADUs per period, each one of length 2 NDUs. Stream 1 delivers variable number of ADUs per period each one of length 1. The system utilisation increases as the number of ADUs for stream 1 increases in steps of five following a uniform distribution. In this case, the difference in both the ADU and the NDU losses is more acute for lower utilisations. The SDS and EDF policies show no losses for utilisations less than approximately 0.95 and lower losses than the SJF policy for a system load between 0.95-1.05. For utilisations above 1.05 the performance curves of all policies tend to converge.

The performance of the SDS policy under multiple, non-synchronised periodic ADU streams as considered in subsection 3.3 (referred to in the figures as SDS-2) was compared to that of EDF policy thought simulations for streams with uniformly shifted arrival instants with respect to each other. Two versions of the EDF policy were considered. One version schedules only complete ADUs and it is called EDFa in this section. The other version serves ADUs either fully or partly according to their length and deadline. This is called EDF in this section.

Fig. 15 shows the ADU losses and NDU losses for $N = 4$ data streams, period $T = 40$, deadlines $D = 40$. For all streams and interarrival time $\tau = 10$ slots. Streams 1,2 and 3 deliver one ADU of length 12 NDUs. Streams 0 delivers one ADU whose length increases -to increase the system load- in steps of four following a uniform distribution. The difference in the ADU loss curves between SDS-2 and the EDF is almost one order of magnitude for system loads higher than 1.0. The performance of EDFa follows that of SDS-2. The NDU losses are approximately the same for all three policies. Similar behavior regarding the ADU and NDU losses has been observed in the results derived for $N = 8, T = 80, D = 80$ with $\tau = 20$.

4 CONCLUSIONS

Two major scheduling policies have been investigated in this paper for real-time (time-sensitive) applications requiring the delivery of application level QoS. Unlike typical (network-level) scheduling policies focusing on Network-level Data Units (NDUs), the proposed policies are driven by Application-level Data Units (ADUs), aiming to maximise the number of complete ADUs delivered on time. The induced ADU - as opposed to NDU - loss probability is adopted as the (application-level) QoS metric of interest and, thus, as a measure of the effectiveness of a policy.

The first ADU-based policies - the FAI-FAO and the partial-FAI-FAO -are based on the application of the Earliest Due First (EDF) principle under periodic ADU arrival streams. These policies are studied analytically and they are shown to outperform the equivalent NDU-based policies. Under the periodic ADU arrival processes considered here - which are typical of continuous media streams - it is shown that the proposed ADU-based policies can be implemented without requiring a complicated searching procedure to determine the earliest expiring ADU but with a complexity similar to that of a FIFO policy.

The second ADU-based policy - the SDS scheduling policy - aims to improve on the performance of the FAI-FAO policy by taking into consideration the ADU sizes in addition to deadlines, while relaxing the periodic ADU arrival structure which has lead to a considerable simplification regarding the study and implementation of the FAI-FAO policy. Unlike the FAI-FAO policy, the SDS policy reconsiders past scheduling decisions for ADUs still present in the system upon new ADU arrival. The developed *scheduling* algorithm of this policy is shown to be optimal, while various effective *service* algorithms have been proposed and investigated whose effectiveness or optimality depends on the ADU arrival process. It has been shown that the SDS policy (employing an ADU-based EDF *service* algorithm) is optimal under certain periodic ADU arrival processes. Finally, the SDS policy has been shown to outperform the ADU-based EDF and the ADU-based Shortest Job First (SJF) policies, whose principles are employed in the design of the SDS policy. A simplified implementation of the SDS policy under periodic ADU arrival processes is presented and is related to the FAI-FAO policy in this environment.

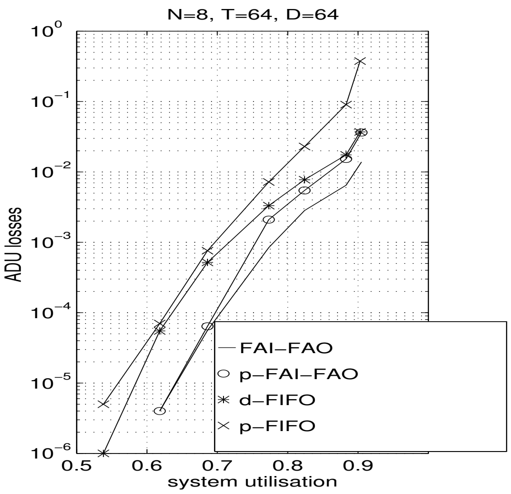

Figure 7: *ADU loss curves for N=8, T=64 and D=64 under geometric arrivals.*

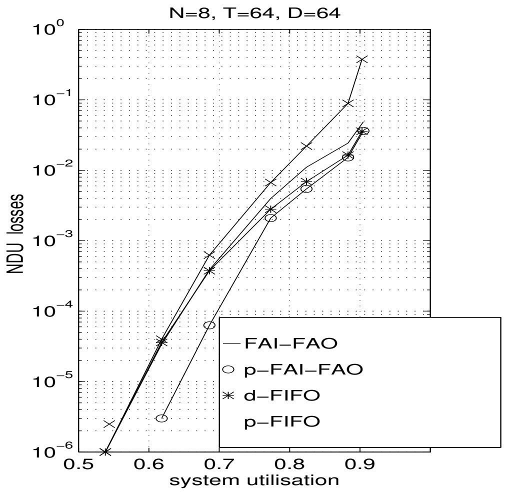

Figure 8: *NDU loss curves for N=8, T=64 and D=64 under geometric arrivals.*

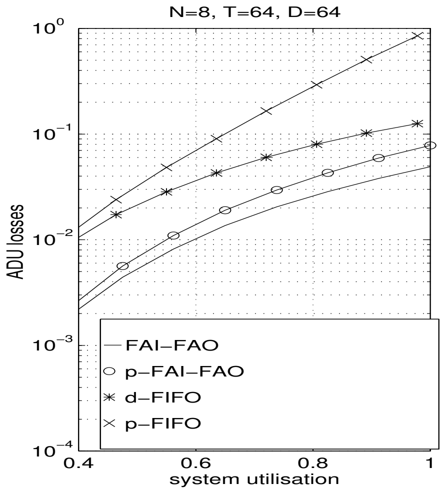

Figure 9: *ADU loss curves for N=8, T=64 and D=64 under bursty 1-L arrivals.*

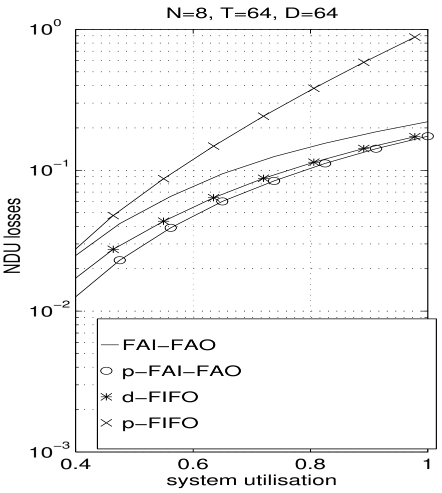

Figure 10: *NDU loss curves for N=8, T=64 and D=64 under bursty 1-L arrivals.*

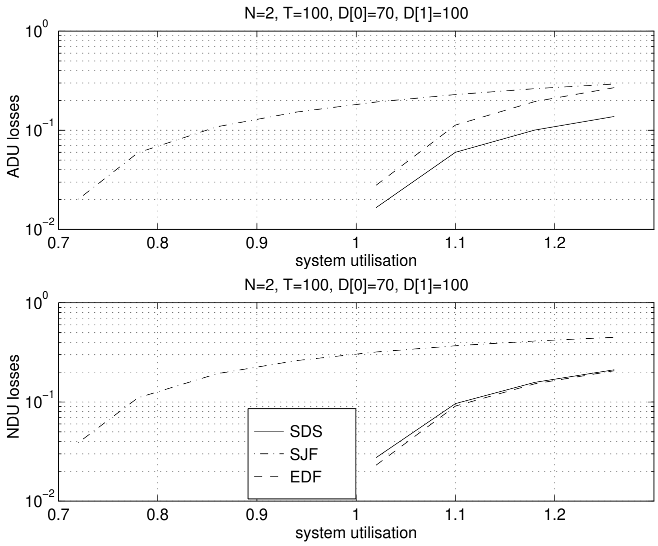

Figure 14: *ADU and NDU loss curves for $N = 2$, $T = 100$, $D^0 = 70$ and $D^1 = 100$.*

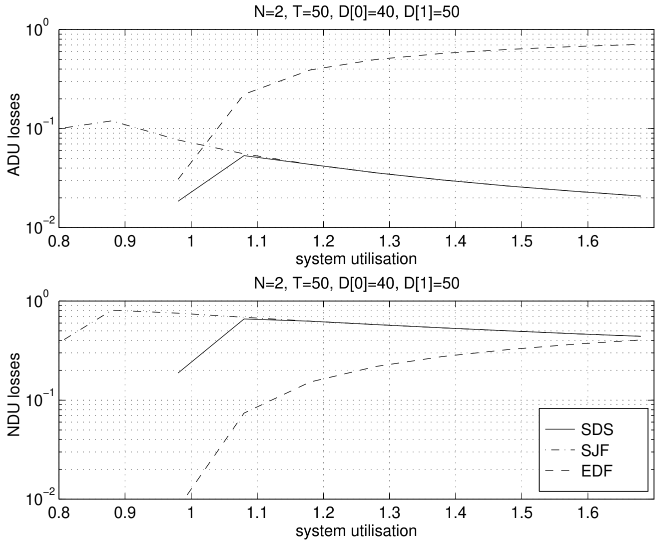

Figure 13: *ADU and NDU loss curves for $N = 2$, $T = 50$, $D^0 = 40$ and $D^1 = 50$.*

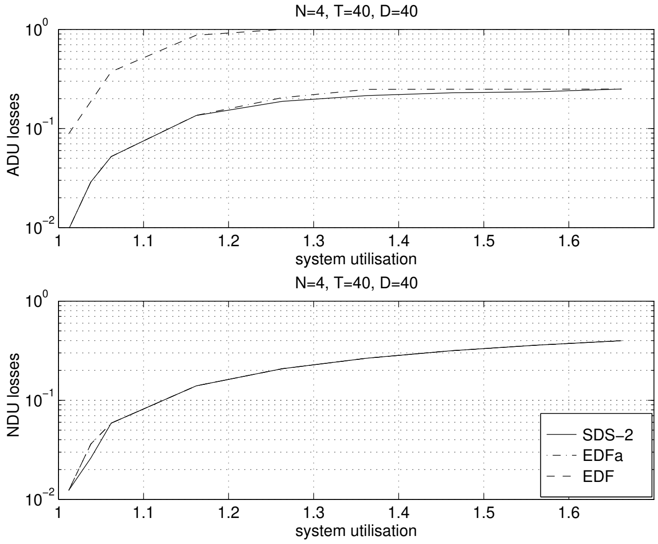

Figure 15: *ADU and NDU loss curves for $N = 4$, $T = 40$, $D = 40$ and $\tau = 10$.*

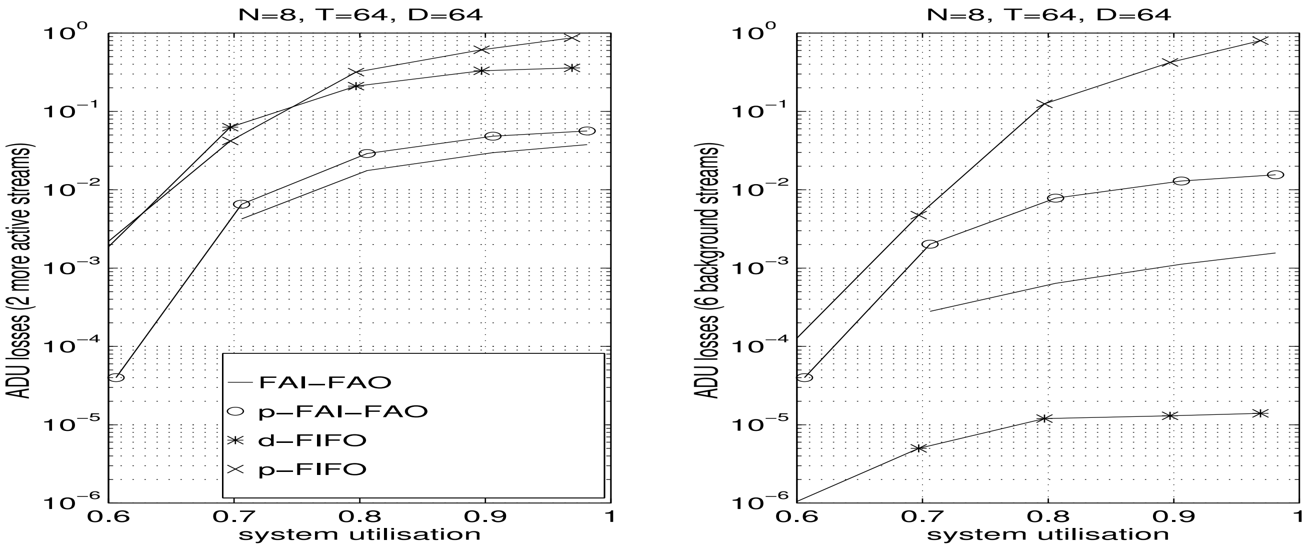

Figure 11: *ADU loss curves for N=8, T=64 and D=64 under heterogeneous bursty 1-L arrivals.*

Appendix

A Performance analysis of the FAI-FAO scheduling policy

Let J_k denote the data stream from which a new ADU arrival occurs at time t_k. The content of the new arrival is to be shifted to the service queue beginning at time t_k. A^{j_k} denotes the random variable representing the number of NDUs from stream j_k that arrived at time t_k. Let Q_k denote the service queue occupancy at time t_k, *after* it has been updated by the number of NDUs in the new ADU from stream J_k. In view of the above, the system process $\{Q_k, J_k\}_{k\geq 0}$ can be defined as a 2-D process with state space $\{(q_k, j_k) : 0 \leq q_k \leq D_{max}, 0 \leq j_k \leq N-1\}$. D_{max} denotes the maximum delay tolerance of all N stream delay tolerances; that is $D_{max} = \max_j\{D^j\}$. Note that for J_k the only transitions allowed are from some j_k to $j_{k+1} = (j_k+1)mod(N)$. Based on the above definition of the system process, it is easy to establish that it is a Markov chain. The remaining of this section contains the derivation of the probability that the Markov chain moves from any state (q_k, j_k) to state (q_{k+1}, j_{k+1}), given by $P(q_k, j_k, q_{k+1}, j_{k+1})$.

A.1 FAI-FAO policy

Under this policy, NDUs of an ADU are served only if the *entire* ADU can meet its deadline. Three distinct cases can be identified, based on the queue occupancy of the service buffer at the instant of a new ADU arrival. In each of the following cases $0 \leq j_k \leq N-1$ and $j_{k+1} = (j_k+1)mod(N)$.

Case A: for $0 \leq q_k \leq \tau_k$. In this case all the NDUs in the buffer will have been served by the time the next ADU arrives from stream j_{k+1} at time t_{k+1} and the new buffer occupancy will be equal to the number of new arrivals only:

$$P(q_k, j_k, q_{k+1}, j_{k+1}) = P\{A^{j_{k+1}} = q_{k+1}\}$$
$$\text{for} \quad 0 \leq q_{k+1} \leq T \tag{2}$$

Case B: for $\tau_k < q_k \leq D^{j_{k+1}} + \tau_k - 1$. In this case by the time the next ADU arrives the service buffer occupancy will have dropped to $(q_k - \tau_k)$. Hence, with the new arrival $A^{j_{k+1}}$ the occupancy will be updated to $(q_k - \tau_k + A^{j_{k+1}})$ if the entire ADU can meet its deadline $D^{j_{k+1}}$:

$$P(q_k, j_k, q_{k+1}, j_{k+1}) = P\{A^{j_{k+1}} = q_{k+1} - (q_k - \tau_k)\}$$
$$\text{for} \quad (q_k - \tau_k) \leq q_{k+1} \leq D^{j_{k+1}} \tag{3}$$

Otherwise, the new ADU will be dropped and the service buffer occupancy will become $(q_k - \tau_k)$:

$$P(q_k, j_k, q_{k+1}, j_{k+1}) = \sum_{n=D^{j_{k+1}}-(q_k-\tau_k)+1}^{T} P\{A^{j_{k+1}} = n\}$$
$$\text{for} \quad q_{k+1} = q_k - \tau_k \tag{4}$$

Case C: for $D^{j_{k+1}} + \tau_k \leq q_k \leq D_{max}$. In this last case the new ADU will definitely miss its deadline and thus it is dropped with probability 1:

$$P(q_k, j_k, q_{k+1}, j_{k+1}) = 1 \quad \text{for} \quad q_{k+1} = q_k - \tau_k \tag{5}$$

A.2 Partial FAI-FAO policy

This policy allows for an ADU to be either partially or fully served. In the event where only a fraction of the ADU can meet its deadline those NDUs are shifted to the service

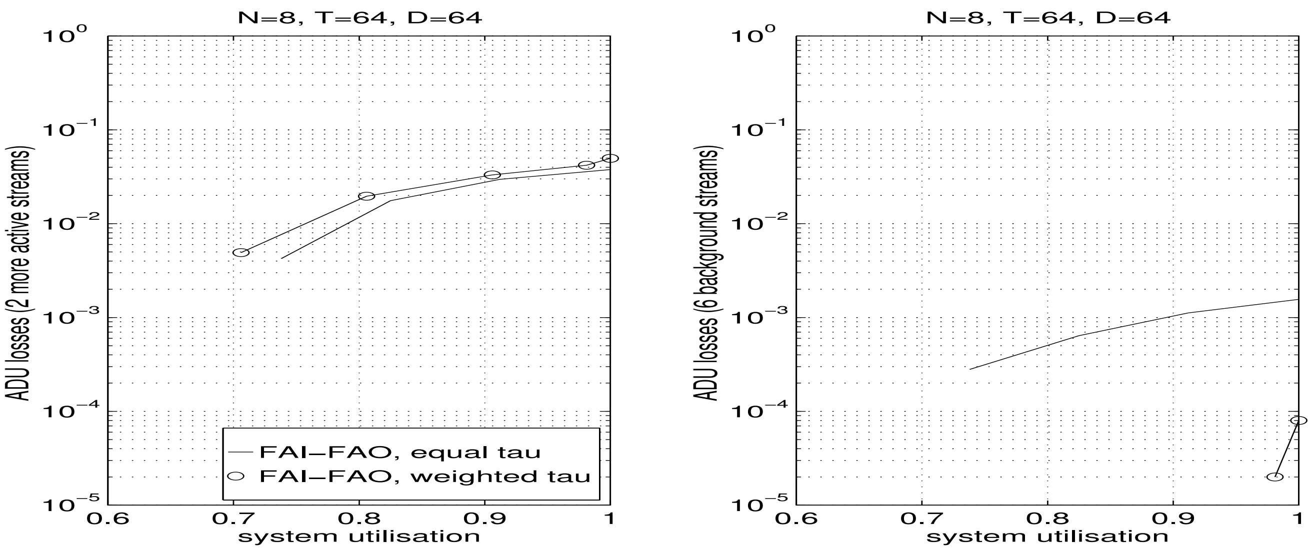

Figure 12: ADU loss curves for N=8, T=64 and D=64 under heterogeneous bursty 1-L arrivals.

buffer and the remaining are dropped. Again three distinct cases can be identified. Cases A and C are the same as in the above section.

Case B: for $\tau_k \leq q_k \leq D_{j_{k+1}} + \tau_k - 1$. In this case depending on the length of the new ADU there are two distinct possibilities of full or partial service:
Full service:

$$P(q_k, j_k, q_{k+1}, j_{k+1}) = P\{A^{j_{k+1}} = q_{k+1} - (q_k - \tau_k)\}$$
$$\text{for} \quad (q_k - \tau_k) \leq q_{k+1} \leq D^{j_{k+1}} - 1 \tag{6}$$

Partial service:

$$P(q_k, j_k, q_{k+1}, j_{k+1}) = \sum_{n=D^{j_{k+1}}-(q_k-\tau_k)}^{T} P\{A^{j_{k+1}} = n\}$$
$$\text{for} \quad q_{k+1} = D^{j_{k+1}} \tag{7}$$

A.3 Computation of ADU Loss and NDU loss probabilities

Apart from the *NDU* loss probability that can be used in order to assess the performance of the proposed policies, another meaningful metric would be the *ADU* loss probability. An ADU is considered to be lost unless all its NDUs are served. The ADU and NDU loss probabilities are easily computed by involving the Markov system process.

A.3.1 ADU loss probability

Since an ADU is considered lost if one of its NDUs is lost, the analysis for the ADU loss probability is the same for both policies.

Let $L^B(q_k, j_{k+1})$ denotes the number of ADUs that are dropped over the interval $[t_{k+1}, t_{k+2})$, where j_{k+1} is the data stream the scheduler will attend to over this interval (following stream j_k) and q_k is the queue occupancy at t_k. Clearly:

$$L^B(q_k, j_{k+1}) = \mathbf{1}_{\{A^{j_{k+1}} > D^{j_{k+1}} - (q_k - \tau_k)\}} \tag{8}$$

The conditional expected value of this indicator function given that the system is in state (q_k, j_{k+1}) is given by:

$$\overline{L^B}(q_k, j_{k+1}) = E\{\mathbf{1}_{\{A^{j_{k+1}} > D^{j_{k+1}} - (q_k - \tau_k)\}}\} \tag{9}$$

It represents the average number of ADUs lost as a function of the queue occupancy at the time instant t_k and the data stream arrival that occurs at t_{k+1} under both policies.

If $0 \leq q_k \leq \tau_k$ then clearly there will be no losses since the server will serve all the NDUs in the service buffer during the interarrival time τ_k between streams j_k and j_{k+1} and thus:

$$L^B(q_k, j_{k+1}) = 0 \text{ and } \overline{L^B}(q_k, j_{k+1}) = 0 \tag{10}$$

If $\tau_k \leq q_k \leq D_{max}$ then losses may occur. The conditional expected value of those losses in any state (q_k, j_{k+1}) is given by:

$$\overline{L^B}(q_k, j_{k+1}) = \sum_{n=D^{j_{k+1}}-(q_k-\tau_k)+1}^{T} P\{A^{j_{k+1}} = n\} \tag{11}$$

Hence, the overall average value of the ADU losses L^B over any interval τ_k will be:

$$\overline{L^B} = \sum_{j=0}^{N-1} \sum_{q_k=\tau_k+1}^{D_{max}} \sum_{n=D^{j_{k+1}}-(q_k-\tau_k)+1}^{T} P\{A^{j_{k+1}} = n\}\pi(q_k, j_k), \tag{12}$$

where $\pi(q_k, j_k)$ is the steady state probability distribution of the corresponding Markov Chain.

Finally, the ADU loss probability is given by:

$$P_B = \frac{E\{\text{ADUs lost over any } \tau_k\}}{E\{\text{ADUs arrived over any } \tau_k\}} = \overline{L^B} \quad (13)$$

The denominator is equal to 1 since by definition of the system, there is only one ADU arrival over any interarrival interval τ_k.

A.3.2 NDU loss probability

The derivation of the NDU loss probabilities is similar for both policies and follows the same procedure as in the above section. Under the FAI-FAO policy, the NDU losses correspond to the NDUs of the ADUs which did not meet the deadline. Whereas under the partial FAI-FAO policy the lost NDUs can be a fraction of the ADU's total number of NDUs.

Again let $L^C(q_k, j_{k+1})$ denote the number of NDUs dropped over the interval $[t_{k+1}, t_{k+2})$, where j_{k+1} is the data stream the scheduler will attend to over this interval and q_k is the queue occupancy at t_k. Also, let $\overline{L^C}(q_k, j_{k+1})$ denote the conditional expected value given that the system is in state (q_k, j_{k+1}).

As in the previous section, if $0 \leq q_k \leq \tau_k$ then clearly there will be no losses since the server will serve all the NDUs in the service buffer during the interarrival time τ_k between streams j_k and j_{k+1} and thus:

$$L^C(q_k, j_{k+1}) = 0 \text{ and } \overline{L^C}(q_k, j_{k+1}) = 0 \quad (14)$$

for both policies.

If $\tau_k \leq q_k \leq D_{max}$ then losses may occur. For the FAI-FAO policy the conditional average number of NDUs lost over any interval τ_k is given by:

$$\overline{L_1^C}(q_k, j_{k+1}) = \sum_{n=D^{j_{k+1}}-(q_k-\tau_k)+1}^{T} nP\{A^{j_{k+1}} = n\} \quad (15)$$

The same quantity for the partial FAI-FAO policy is given by:

$$\overline{L_2^C}(q_k, j_{k+1}) = \sum_{n=D^{j_{k+1}}-(q_k-\tau_k)+1}^{T} [n-(D_{j_{k+1}}-(q_k-\tau_k))]P\{A^{j_{k+1}} = n\} \quad (16)$$

The overall average values of $L_1^C(q_k, j_{k+1})$ and $L_2^C(q_k, j_{k+1})$ is given by:

$$\overline{L_1^C}(q_k, j_{k+1}) = \sum_{j=0}^{N-1} \sum_{q_k=\tau_k+1}^{D_{max}} \sum_{n=D^{j_{k+1}}-(q_k-\tau_k)+1}^{T} nP\{A^{j_{k+1}} = n\}\pi(q_k, j_k) \quad (17)$$

$$\overline{L_2^C}(q_k, j_{k+1}) = \sum_{j=0}^{N-1} \sum_{q_k=\tau_k+1}^{D_{max}} \sum_{n=D^{j_{k+1}}-(q_k-\tau_k)+1}^{T} \ldots$$

$$P\{A^{j_{k+1}} = n\}[n-(D^{j_{k+1}}-(q_k-\tau_k))]\pi(q_k, j_k), \quad (18)$$

where $\pi(q_k, j_k)$ is the steady state probability distribution of the corresponding Markov Chain.

Finally, the NDU loss probability is given by:

$$P_C = \frac{E\{\text{NDUs lost over any } \tau_k\}}{E\{\text{NDUs arrived over any } \tau_k\}} \quad (19)$$

The numerator is given by Equations (17) and (18). The denominator represents the average number of NDUs that arrive in the system over any interval τ_k. It can be computed by summing the mean ADU length of each data stream weighted by the probability that an ADU that arrives in any given interval τ_k belongs to a particular data streams. Since all the data streams have the same frequency of ADU arrivals in a period the weighted sum over any interval τ_k is given by:

$$\overline{A} = \sum_{j=0}^{N-1} \frac{1}{N}\overline{A^j} \quad (20)$$

B THE SERVICE ALGORITHM

In this appendix the combined SJF/EDF service algorithm is described. Since the service algorithm determines the ADUs which will still be in the system when a future arrival occurs (next decision time) it affects the ADU dropping process at that time and, thus, the resulting performance of the proposed SDS scheduling policy. An algorithm which could guarantee that in any given time interval the number of served ADUs is the maximum possible is the SJF algorithm. However, the SJF approach can not be used in this case due to the diverse deadlines of the scheduled ADUs. Therefore, an alternative approach has been developed. This approach attempts to serve as many of the scheduled ADUs as possible using the SJF algorithm without causing any of the remaining ADUs to violate their deadlines. The remaining of the ADUs are served according to 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 ADUs have been scheduled and their deadlines. Let t denote the current decision time instant. Let $\mathcal{P}_t^s = \{P_1, P_2, ..., P_n\}$ denote the set of scheduled ADUs 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 ADUs 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 ADU first (P) and examines whether any of the ADUs with earlier deadlines (shorter counter values, I_i) will violate them if ADU P were to be served first followed by the remaining ADUs served in an EDF fashion (step 2). There is no need to check the ADUs with later deadlines since it is guaranteed that there are sufficient resources to serve all scheduled ADUs using the EDF algorithm. If there is no deadline violation, ADU P is served and the delay tolerances (counters) are reduced by an amount equal to its length $\hat{P}$ (step 2-(a)). If a possible deadline violation is detected, though, ADU P is not served. Instead, the server will consider those ADUs whose deadline would have been violated. Again, it will start from the shortest ADU and examine whether any violations occur to ADUs of earlier deadlines (if any). If not, it will proceed to serve those ADUs fully (step 2-(b)). Finally, the updated value of the counters is kept to be used as its initial value when the next ADU is considered (end of step 2-(a)).

At the arrival (decision) time t:

1. Create $\mathcal{P}_t^s$; that is order the ADUs in the system according to SJF. Create $\mathcal{T}_i$. Initialise counters I_i, $1 \le i \le k$. $I_l(0) = I_l$, $1 \le l \le k$. Select the shortest ADU, say P, in $\mathcal{P}_t^s$.

2. Suppose $P \in \mathcal{T}_i$ with W_i. In order to check for a possible deadline violation the following conditions are checked: compute the sums $\sum_{l=1}^{m} W_l \le I_m - \hat{P}$ for $1 \le m < i$.

   (a) If all inequalities hold true set $I_l(0) = I_l$ for $1 \le l \le k$; set $W_i(0) = W_i$; set $m = 1$ and repeat for $\hat{P}$ steps while ADU is being served:

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

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

      Set $I_l = I_l(\hat{P})$; set $W_i = W_i(\hat{P})$; remove P from $\mathcal{P}_t^s$.

   (b) If the inequality does not hold true for the ADUs belonging to some set then a possible deadline violation has been detected. Then select the shortest of those ADUs, say P. Go to the beginning of step 2.

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

The following example illustrates the above procedure. Suppose there are two streams with $D^1 = 16$ slots and $D^2 = 22$ slots. Each has two ADUs to be served; the ADUs are numbered from 1 to 4 according to their size. So $\mathcal{P}_t = \{P_1, P_2, P_3, P_4\}$, with $\hat{P}_1 = 2$, $\hat{P}_2 = 3$, $\hat{P}_3 = 5$, $\hat{P}_4 = 6$ slots. $P_1, P_2 \in C^2$ and $P_3, P_4 \in C^1$. Also, $W_1 = 5 + 6 = 11$ slots and $W_2 = 2 + 4 = 6$ slots. Initially, counter I_1 is set to D^1 and counter I_2 is set to D^2.

If the EDF algorithm were to be used all the ADUs can meet their deadline; ADUs P_3 and P_4 would be served first followed by P_1 and P_2. The above procedure though would attempt to start from ADU P_1. It would first check if the ADUs of stream 1 will violate their deadline: $W_1 < I_1 - \hat{P}_1 \rightarrow 11 < 16 - 2 = 14$. Since no deadline is violated ADU P_1 is served and the counters are updated as follows: $I_1 = I_1 - 2 = 14$, $I_2 = I_2 - 2 = 20$ and $W_2 = W_2 - 2 = 4$ slots. Next, it would attempt to serve ADU P_2. However, since the condition $W_1 < I_1 - \hat{P}_2 \rightarrow 11 < 14 - 4 = 10$ is not met and the ADUs of stream 1 will miss their deadline, the algorithm would now focus on stream 1, serve ADUs P_3 and P_4 and then it would complete by serving ADU P_2.

C Optimality of the SDS policy

The function of the SDS scheduling algorithm is to construct the set of ADUs that can be served, $\mathcal{P}_t^s$, out of the set of ADUs present in the system, $\mathcal{P}_t$, at any decision instant t. For the environment considered in this paper, it is shown that the SDS scheduling algorithm determines the maximum number of ADUs that can be scheduled and *no* modification to $\mathcal{P}_t^s$ can increase its size.

The proof is shown here for two ADU traffic classes C^1 and C^2 with deadlines D^1, D^2 and associated counters I^1 and, respectively, I^2. Without loss of generality it is assumed that $D^1 < D^2$ which implies that at all times $I^1 \le I^2$. It can be extended to any number of classes.

Let $\mathcal{P}_t = \{P_1, P_2, ..., P_M\}$ and $\mathcal{P}_t^s = \{P_{j_1}, P_{j_2}, ..., P_j, ..., P_J\}$ where both sets are ordered in terms of increasing sizes. Let P_j denote the last ADU (from any of the two classes) that was scheduled for service before the first ADU, P_{j+1}, was dropped (from any class). Since P_{j+1} is the shortest unscheduled ADU, it will require the least amount of resources (reduction to counters) than any other unscheduled ADU. The proof below shows that, P_{j+1} can be accommodated at best, by pushing out another ADU, P, from $\mathcal{P}_t^s$. That is, it will not increase the number of scheduled ADUs. Consequently, no other unscheduled ADU -which would require more resources than P_{j+1}- can do it as well. Two distinct cases are considered:

Case A: $P_{j+1} \in C^2$. Since P_{j+1} is dropped its size $\hat{P}_{j+1}$ satisfies $\hat{P}_{j+1} > I^2 \le I^1$. Notice that if $P_{j+1} \in C^2$ then $\mathcal{P}_t^s = \{\text{J shortest ADUs of } \mathcal{P}_t\}$ which, in turn, implies that $P_{j+1} > P$. It is, therefore, expected to be the longest possible set of scheduled ADUs. Consider two cases:

1. Assume that $P \in C^2$. By pushing out P counter I^2 will be updated to: $I'^2 = I^2 + \hat{P}$. P_{j+1} can be scheduled only if $I'^2 \ge \hat{P}_{j+1}$ in which case P is removed from $\mathcal{P}_t^s$ and P_{j+1} is added resulting in the same number of scheduled ADUs as before.[2] Increasing the size of $\mathcal{P}_t^s$ has most chances of being accomplished by considering the shortest unscheduled

[2]Else, two previously scheduled ADUs need to be pushed out in order to accommodate P_{j+1}. The same arguments as above can be followed for this case.

ADU for service, which is P. This would be possible only if $I'^2 - \hat{P}_{j+1} \geq P \Rightarrow I^2 + P - \hat{P}_{j+1} \geq P \Rightarrow I^2 \geq \hat{P}_{j+1}$ which is not true. So the size of $\mathcal{P}_t^s$ can not increase by removing an ADU from C^2 to accommodate P_{j+1}.

2. The same argument can be followed if $P \in C^1$, because $I^1 \leq I^2 < \hat{P}_{j+1}$.

Case B: $P_{j+1} \in C^1$. Since P_{j+1} is dropped its size $\hat{P}_{j+1}$ satisfies $\hat{P}_{j+1} > I^1$. Consider two cases:

1. No ADU was scheduled after P_j. This implies that $P \leq P_{j+1}$. Consider two distinct cases:
   (a) Assume that $P \in C^1$. By pushing out P counter I^1 will be updated to: $I'^1 = I^1 + \hat{P}$. P_{j+1} can be scheduled only f $I'^1 \geq \hat{P}_{j+1}$ in which case P is removed from $\mathcal{P}_t^s$ and P_{j+1} is added resulting in the same number of scheduled ADUs as before. Increasing the size of $\mathcal{P}_t^s$ has most chances of being accomplished by considering the shortest unscheduled ADU for service, which is P. This would be possible only if $I'^1 - \hat{P}_{j+1} \geq P \Rightarrow I^1 + P - \hat{P}_{j+1} \geq P \Rightarrow I^1 \geq \hat{P}_{j+1}$ which is not true. So the size of $\mathcal{P}_t^s$ can not increase by removing an ADU from C^1 to accommodate P_{j+1}.
   (b) Assume that $P \in C^2$. In this cases, consider the two possibilities:
      i. $I^1 = I^2$: which leads to the same conclusion as above by following the same argument.
      ii. $I^1 < I^2$: in which case the originally scheduled P has not consumed any legitimate time from class C^1 ADUs and removing it from the set of scheduled ADUs will not affect counter I^1. ADUs of class C^2 consume legitimate time from ADUs of class C^1 from the instant (slots) when I^2 becomes equal to I^1.

2. At least one ADU from C^2 was scheduled following the drop of P_{j+1}. It will be shown that even if P is the largest scheduled ADU, P_{j+1} can not be accommodated. If P is the largest scheduled ADU then $P \in C^2$ and $P \geq P_{j+1}$. This implies that counter I^2 satisfied: $I^2 > I^1$ at the time P was scheduled. So by pushing out P counter I^1 will be updated, at best, to the value it had at the time P_{j+1} was originally rejected. So pushing out P does not accommodate P_{j+1} and the size of $\mathcal{P}_t^s$ can only decrease.

REFERENCES

[1] Klara Nahrstedt, Ian R. Philip 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] George Kesidis. *ATM Network Performance*. Kluwer Academic Publisher, 1996.

[3] R.P. Tsang et al. Dynamic resource control for continuous media traffic over atm networks. *Computer Communications, v. 19, pp. 1092-1111*, 1996.

[4] K. Nahrstedt. Challenges of provided end-to-end qos guarantees in networked multimedia systems. *Technical Report, University of Illinois at Urbana-Champaign, Ill.*

[5] V. Srinivasan, A. Ghanwani and E. Gelenbe. Block loss reduction in atm networks. *Computer Communications, v. 19, pp. 1077-1091*, 1996.

[6] J.M. Mc Manus and K.W. Ross. Video on demand over atm: Constant-rate transmission and transport. *Technical Report, University of Pennsylvania, Philadelphia, PA, http://www.seas.upenn.edu/∼ross/*, November 1995.

[7] Marcel Graf. VBR Video over ATM: Reducing Network Resource Requirements through Endsystem Traffic Shaping. *Proceedings of IEEE Infocom*, 1997.

[8] 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.

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

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

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

[12] X. Wang and I. Stavrakakis. Study of scheduling for group-based quality of service delivery. *Third volume on Performance Modeling and Evaluation of ATM Networks, ed.: D. Kouvatsos, Chapman & Hall*, 1997.

[13] A.F. Dagiuklas and M. Ghanbari. Effect of video synchronisation in an ATM multiplexer. *IEE Electronic Letters*, January 1994.

[14] Byeong-Hee Roh, Hee-June Ahn and Jae-Kyoon Kim. A Starting Time Selection Method for Minimum Cell Loss Ratio of Superimposed VBR MPEG Video Traffic in ATM Networks. *Technical Report, KAIST, Korea*, 1997.