Performance Evaluation of the Deadline Credit Scheduling Algorithm for Soft-Real-Time Applications in Distributed Video-on-Demand Systems

Adamantia Alexandraki, Michael Paterakis

Technical University of Crete
Department of Electronics & Computer Engineering
Laboratory of Information & Computer Networks
GR-731-00 Chania, Greece
e-mail: { adalex,pateraki } @telecom.tuc.gr

Abstract: Video-on-Demand (VoD) systems are expected to support a variety of multimedia services to the users, such as tele-education, teleconference, remote working, videotelephony, high-definition TV, etc. These applications necessitate abundant bandwidth and buffer space as well as appropriate software and hardware for the efficient manipulation of the network's resources.

In this work we investigate a promising scheduling algorithm referred to as the Deadline Credit (DC) algorithm, which exploits the available bandwidth and buffer space to serve a diverse class of prerecorded video applications. We provide simulation results when the DC algorithm is applied to a hierarchical architecture distributed VoD network, which fits the existing tree topology used in today's cable TV systems. The issues investigated via the simulations are: the system utilization, the influence of the buffer space on the delivered Quality of Service, and the fairness of the scheduling mechanism. We examine cases with homogenous as well as diverse video streams, and extend our system to support interactive VCR-like functions. We also contribute a modification to the DC algorithm so that in cases when the video applications have different displaying periods, the video streams obtain a fair share of the network's resources. Finally, we validate our results by simulating actual videos encoded in MPEG-4 and H.263 formats.

Keywords: Scheduling video applications, distributed VoD systems, MPEG-4, H.263, VCR-like operations.

1 Introduction

Telecommunication networks have lately extended to support a variety of services to the users such as tele-education, teleconference, remote working, videotelephony, high-definition TV, web video streaming, etc. These applications dictate the need for high-speed networks, and also necessitate the appropriate software and hardware for the efficient integration of a diverse class of applications with different characteristics. These demanding requirements led to the concept of ***broadband integrated digital networks (B-ISDN)***, which are systems of high capabilities that enable the transmission of several different applications.

Several research areas have been developed in order to increase the efficiency of the network and keep the Q*uality of Services (QoS)* at a satisfactory level. One significant area that affects the quality

provided by the network is the ***scheduling process***. Since several applications should be concurrently serviced, the role of the scheduling process is to select each time the application that should be served next.

The applications that telecommunication networks support are generally classified into two categories based on the delay that they can undergo during connection: the *real time* and the *non-real time* applications. Real time applications involve a kind of interactivity and consequently can tolerate no delay. The characteristic of these applications is that the network's resources which provide the services need to be reserved for the duration of the connection. This necessitates ample bandwidth that is adequate to support the real time applications with the required QoS. Non-real time applications, on the other hand, can tolerate finite delays, but they require large buffer space in order to be stored during their service. Video-on-Demand (VoD) applications, which is the focus of this paper, can be classified somewhere between real and non-real time applications and they are usually referred as ***soft real-time applications***. When video is displayed, certain time constraints need to be met, i.e. the consequent scenes in the video should reach the receiving end during finite time intervals. This brings video closer to real time applications. However, it is not necessary that the video scenes are generated in real time; they can be recorded *a priori*, but this entails the need for adequate buffer space. For these characteristics of the video applications, in order to achieve the desired QoS, we need to develop systems that combine high bandwidth with large buffer space and achieve efficient manipulation of these resources.

In this work, we use the Asynchronous Transfer Mode (ATM) technique of switching and multiplexing multimedia streams, which encapsulates traffic into cells that travel along the network's links [4], [19]. A video application provides the network with a number of video frames at a rate that is referred as *frame rate*. This is the rate at which an image is digitized, compressed and cut to fit into ATM cells. It is also the rate at which frames should be available at the receiving end for the decoding and reconstruction processes. This periodicity as well as the video's frame size variability, should be taken into consideration in order to guarantee the arrival of frames within time constraints and the simultaneous avoidance of overflow at the receiving points.

A B-ISDN supports a diverse class of applications with different constant or variable bandwidth requirements. In this work we use VBR-encoded video, instead of CBR video, because for the same image quality VBR-encoded video can have a significantly lower average bit rate as well as exhibit considerable multiplexing gain.

We evaluate through simulation the performance of an innovative scheduling scheme referred to as the Deadline Credit Algorithm, originally proposed in [1] and [3]. We use the term ***application data unit*** (***ADU***), which refers to the ATM cells that belong to the same video frame. Each ADU has a sequence number (SN) indicating the scene that it belongs to. As a metric for performance

evaluation the authors in [1] use the ***application data unit loss rate*** instead of the *cell loss rate,* since it is assumed that an image is degraded even if one of its cells is lost. We examine the DC algorithm and the role that buffers can play in increasing its efficiency. We consider only cases with small buffer availability because we intend to show that DC works quite well even when storage is a restricted parameter. Finally, we investigate the node utilization of the system, and we concentrate on evaluating the fairness of the scheduling process.

In section 1.1, we provide a brief overview of the DC algorithm. In section 2, we describe the distributed VoD system to be used in our study. Section 3 contains the simulation results for this system when the DC algorithm is applied. We examine cases with homogenous as well as diverse video streams, and expand our system to support the basic interactive functions (VCR-like functions). In section 4, we present simulation results when actual prerecorded MPEG-4 and H.263 video programs are used, and finally, in section 5, we present the conclusions of our work.

1.1 DC Algorithm Overview

There are frequent periods of time during which the network's bandwidth is under-utilized. During these periods, frames from the ongoing videos can be pre-fetched at destination ends or somewhere near. This can alleviate traffic at times of network's over-utilization, but requires an adequate storage capacity to keep the frames until they can be processed. In order to treat video streams fairly, the number of in advance forwarded ADUs should be equal among the contending streams and no stream should monopolize the network's resources. DC tries to forward an equal number of ADUs from each traffic flow and keep the streams at the same level of delay allowance. In addition, DC is a real-time reactive algorithm when the system is unable to support the stream's bandwidth demands. When the network is overloaded and as a consequence frames need to be dropped, DC tries to distribute the losses fairly across streams. The algorithm's philosophy is that if some ADUs of a stream are dropped this gives priority to the stream for future ADU transmissions compared to streams with less or no losses.

The time within which an ADU has to reach its destination node can be spent in a variety of ways on the traversing nodes. The DC algorithm distributes this time over the nodes that an ADU traverses and poses restrictions on the time of departure from these nodes. Particularly, it poses an upper bound on the time when an ADU from stream j should have left the corresponding node in order to arrive at the next node within time constraints. Consequently, if the ADU cannot arrive in time at the next node, it is dropped, whereas, if an ADU is served, it is always guaranteed that it will arrive at the next node in time. The deadline is defined cumulatively from one node to the next, and if an ADU leaves a node before its deadline, the remaining slots of its deadline can be spent at the next nodes.

Each time a stream should be selected for transmission, the DC algorithm selects the stream with the maximum number of ADU losses. Among streams with the same number of ADU losses the DC algorithm selects the stream with the minimum allowable delay, i.e. it serves the ADU that can be delayed the minimum time at the corresponding node. This is expressed in the priority index counter (P_CR), which is used to select between the contending streams the stream that should be served next. Let the superscript *j* denote the stream and the subscript *n* denote the node. Each stream at each node has a P_CR defined as follows:

$$\mathrm{P\_CR}_n^j = \frac{D\_CR_n^j - T^j * L\_CR_n^j}{T^j},$$

where, the losses counter ($L\_CR_n^j$) keeps information on the number of ADU losses that stream j has suffered from previous nodes up to node *n* and the deadline credit counter ($D\_CR_n^j$, measured in slots) expresses the time that the ADU at the head of the queue of the stream *j* can be delayed at node *n*. The term T^j refers to the period of the stream. The smaller the value of the priority index counter of a stream, the higher the priority of the stream with respect to that of the other stream

However, the buffer space at the traversing nodes is limited so before an ADU is transmitted the algorithm has to guarantee that there will not be buffer overflow at down stream nodes. In this work, we avoid buffer overflow using the buffer counter ($Buffer_n^j$)[1]. $Buffer_n^j$ is initialised one unit at node *n* for every ADU of maximum length from stream *j* it can hold. Whenever an ADU of stream j arrives at node n or whenever the transmission of an ADU from stream j, already stored in the corresponding buffer at node n, starts, the buffer counter is decreased or increased by one, respectively. Although, feedback for the exact free space of each node could be sent, we chose for simplicity to treat all ADUs as if they were of the maximum length. We assume that node *n* informs node *n-1* about its buffer condition, in order that node *n-1* does not transmit an ADU that will cause buffer overflow at node *n*. In contrast to [1], where the feedback is available periodically, in this work it is assumed that feedback is sent when an ADU is transmitted and the buffer occupancy changes. In this case, previous nodes always know the exact buffer state at next nodes and additional unnecessary feedback is avoided. Moreover, since information is sent from node *n* at the beginning of the ADU's transmission to node *n-1*, the next ADU from node *n-1* can be transmitted as soon as there is adequate space at node *n* to store it. Thus, we achieve concurrent transmissions and increased system utilization.

In conclusion, the DC algorithm aims to: (a) maximize the number of ADUs served by giving priority to the ADU with the shortest time to expiration, (b) to send as many ADUs as possible from

[1] The original DC algorithm uses an upper bound on the D_CR counter to avoid buffer overflow. However, we found that this part of the algorithm does not function properly, therefore, the introduction of the buffer counter.

each video stream as closer as possible to the destination node, and (c) to be fair relatively to ADU losses. It is an ADU driven policy that determines whether to transmit or drop entire ADUs.

2 Video-On-Demand System

2.1 Introduction

Several years ago, the concept of a computer network was related to a single central unit that was able to serve all the applications. Networks were centralized and all the users must have had access to the main server in order to obtain the information they required. However, this idea was soon abandoned, primarily due to the increasing complexity of the large centralized servers. The rapid increase in the number of applications and users dictated the need for distributed processing, which was later to become the standard method of data processing in most of today's networks.

Distributed information systems are now able to support several kinds of applications. They are widely used for data, voice and video transmission. Particularly, for video applications, the case we are interested in, several network architectures have been developed that enable users to have access to a wide range of video programs on demand. In the following sections we examine a hierarchical architecture for a VoD distributed network, which fits the existing tree topology used in today's cable TV systems [2], [18], [20], [21].

2.2 System Definition

The distribution network topology is considered to be a binary tree associating a number of video servers and switching nodes connected via communication links. As shown in figure 2.2, at each node $N_{i,j}$ there is a storage device where some prerecorded video streams $X_{i,j,k}$ are stored. In addition, there is a video server $VS_{i,j}$ associated with each switching node $N_{i,j}$, which supplies children nodes with the video streams that are prerecorded at node $N_{i,j}$, as well as with those that arrive at $N_{i,j}$ from its father stream node. A node selects a stream for transmission, according to a scheduling process (DC algorithm in our experiments), and starts transmitting it to the corresponding children node.

A tree network of depth **Dep** consists of 2^{Dep}-1 nodes and each level **L** ($1 \le L \le Dep$) of the tree has 2^{L-1} nodes. The nodes of the **1st**, **2nd**, …, (**Dep-1**)th level run the DC algorithm for the streams they serve. The nodes of the last level (**Dep**th level), consume the arriving ADUs as well as the ADUs that are prerecorded at these nodes. The prerecorded video streams stored at these nodes have no losses, since the users have direct access to the last level of the tree topology. In our experiments we do not take into account these streams, as they do not affect the performance of the DC algorithm.

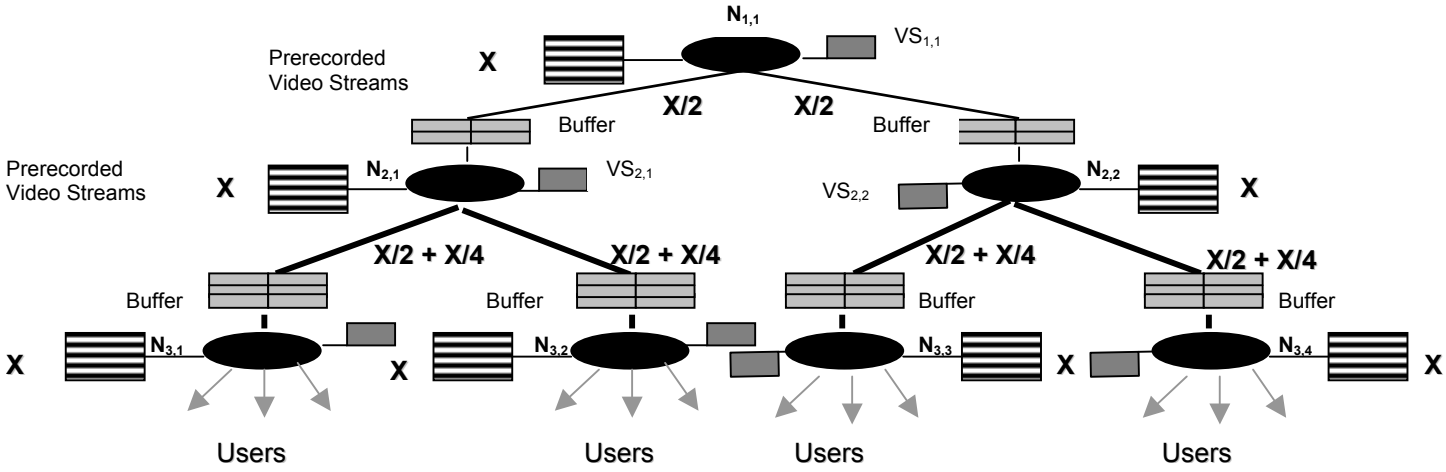

Figure 2.2: *A binary tree network of 3 levels. At each node there are X prerecorded video streams as well as some buffer space for the arriving streams. Users are connected to the headends.*

All the users are connected to the leaf nodes, also called headends. When users require a video program, they connect to the appropriate headend and a request is sent cumulatively upwards through the traversing nodes until it reaches the source node. As soon as the source is informed, the connection is established and the video stream starts to travel along the appropriate path to reach the destination node.

We assume a slotted system, where the slot duration is equal to the time that it takes for an ATM cell to be transmitted. The links between the nodes are supposed to be bi-directional. Video streams travel along one direction (from their storage servers to the requesting clients) and information about the next nodes' condition, for overflow avoidance purposes, goes along the opposite direction. The streams are served in a kind of store-and-forward way by the intermediate nodes. At each system node there is some buffer space for the arriving streams, where they are stored until they can be transmitted by the node. This buffer space is supposed to be limited and its size is a parameter of the system. Each arriving stream is supposed to have its own available buffer space**;** for the DC algorithm, this is denoted by the parameter Buffer_n^j for the stream j at node n.

What usually happens in such networks is that the video program's popularity determines its placement. The most popular programs are stored near the users, whereas the least popular are stored at the upper levels of the tree [2], [18]. The nodes of the same tree level are assumed to have identical prerecorded video streams (duplicate placement), so that a great number of users can be served, and all users can have access to all of the available video programs.

We assume that the system does not support parallel transmission toward the two children stream nodes; all the streams that a node serves are time multiplexed, and each node transmits only to one of its children nodes at a time. If we enabled parallel transmission, then we would have fewer ADU

losses, but the system would be more complex, since a scheduling process would have to be applied at each output link.

Although there are many options relatively to the number of video programs stored at each node, in this work, we assume that the video streams are distributed equally to the network's levels. We have chosen that all the network nodes have an equal number of prerecorded video programs (i.e. $X_{i,j} = \mathbf{X}$, $\forall$ i, j). **X** is another parameter of the system in the experiments we carry out. Furthermore, in our experiments all the prerecorded streams are considered "active". This means that the level a node belongs determines the number of streams it serves, i.e. down stream nodes have more streams to serve than up stream nodes. As far as the buffer space is concerned notice that as we move down the tree, since the number of arriving streams increases, so does the total available buffer space.

Finally, we assume that the users' requests are uniformly distributed among video programs and that each video stream has been generated by a single headend request. Specifically, we assume that the prerecorded video streams of a node as well as the streams that arrive from its father node are distributed equally to its children nodes with the intention that the nodes of the same level are similarly loaded. Accordingly, the number of video streams served by a node depends on the node's level. For a node at the n^{th} level ($n \geq 1$), there are $\sum_{k=1}^{n} \frac{X}{2^{k-1}}$ $(= X * \sum_{k=1}^{n} \frac{1}{2^{k-1}})$ streams served and $\sum_{k=1}^{n} \frac{X}{2^{k}} (= X * \sum_{k=1}^{n} \frac{1}{2^{k}})$ streams that travel to each children node. Notice that the number of the served streams depends on the number of the arriving streams, which depends on the level that a node belongs, and the number of the prerecorded streams at each node (i.e., **X**). The number of the served streams increases as we move down the tree, thus the nearer a network node to the destination nodes, the more loaded the node will be. Since our main goal remains to evaluate the fairness of the DC algorithm, the assumption that each program is required by a single headend does not affect the results. If some videos were requested by more than one headend, this would increase the number of served video streams and the load of the traversing nodes; however, it would not affect the performance of the scheduling process. The only difference would arise in the number of ADU losses and the utilization of the nodes, while the selection process would not differ.

In the following experiments we assume the distributed video-on-demand system of Fig.2.2 with depth three. We are interested in evaluating the ratio of ADU losses, the nodes' utilization and the distribution of the losses across the contending streams. The results shown below have been computed as average values from at least 10 replicated experiments (Monte Carlo Simulation).

3 Performance Evaluation Experiments

3.1 Scheduling Homogenous Streams

The aim of the first experiment is to investigate how the DC algorithm works for a distributed system when homogenous streams are served.

3.1.1 Experiment

All the streams are supposed to have ADUs with uniform length distribution in the interval [1,10] and period equal to T^j = 100 slots[2]. The length of an ADU is independent of the length of the preceding and following ADUs. At destination nodes the ADUs are consumed one per T^j slots from the beginning of the session of the stream. We assume that the sessions of all streams start at time instant 0, so that the first ADU from each stream will be required at destination node at time instant T^j, the second at $2*T^j$, and so on. If a required ADU is stored in the buffer, it is consumed. Otherwise, it is assumed to be lost and the L_CR counter is updated. If an ADU with greater SN than the required is stored in buffer, we do not take it but we leave it until it is required.

We have simulated for a number of 300,000 ADUs for each stream, which, for a frame rate equal to 30 frames/second, corresponds to about 2.8 hours of system operation.

3.1.2 Results and Discussion

In Figure 3.1.2.a we show the ratio of ADU losses as a function of the number of prerecorded streams stored at each node (**X)**. These results have been obtained by dividing the ADU losses of each stream to the total number of required ADUs. We assume that all the prerecorded streams are requested; consequently, as **X** increases, the number of multiplexed streams at each node increases as well. This is also shown in the Figure. Notice that as **X** increases, so does the total number of ADU losses.

The ratio of ADU losses depends on buffer availability only for a small number of multiplexed streams, when the load of the nodes is low and the streams can build a significant deadline credit (**X**<16). On the contrary, when a great number of streams are multiplexed, the nodes are overloaded and the buffer plays no role. In these cases, the system is overutilized and no ADUs are transmitted in advance. Instead, several losses happen due to bandwidth restrictions; the system cannot transmit all the contending ADUs, and consequently the buffer cannot be exploited (**X**≥16).

In Figures 3.1.2.b and 3.1.2.c, the utilizations of the nodes at the first and second level of the tree are shown. Notice that the utilization of a node depends on the level it belongs to. As we move down

[2] We have also simulated two additional scenarios in which the streams have ADUs with uniform length distribution in the intervals [1,5] and [1,20], respectively. However, due to lack of space we omit these results. The conclusions from these experiments are similar to those of the experiment we present in this section.

the tree, nodes have to serve more traffic streams and thus their load increases. Nodes of the same level have to serve the same number of streams and as a result have similar utilization. Comparing these two graphs we notice an increase in utilization of about 50% in the second level for the same **X** value. This is the consequence of the fact that the second level of the tree has 50% (**X**/2) more streams to serve than the first level.

Referring to these Figures, we note that the utilization increases as the number of multiplexed streams increases because more streams need to be served. For small values of **X**, we observe only a minimal number of losses (see Figure 3.1.2.a). In this case, the nodes take advantage of buffer space to send ADUs in advance. When the losses increase, the curves converge at 100% utilization because the nodes become overloaded.

Furthermore, we notice an increase in the nodes' utilization when the available buffer increases. In both cases, the difference is significant when comparing values for buffer=1 ADU and buffer=2 or 5 ADUs. On the other hand, we do not observe discernable difference for buffer=2 and 5 ADUs, because in this case the parent node may send ADUs in advance to children-nodes but utilization remains low since these ADUs are blocked at children-nodes because the latter nodes are more utilized than the parent node. If the goal is to achieve high node utilizations, then a greater buffer should be used. For a substantially deep tree, an option could be to distribute available buffer unequally among nodes. We could increase buffer space as we go up the tree enabling the upper nodes, which are less loaded, to transmit several ADUs in advance and increase utilization.

In Figure 3.1.2.d we show how the ADU losses are distributed across streams as observed from the 2nd level of the hierarchical structure of the tree network. Y-axis shows the ratio of ADU losses and the x-axis the stream index. Streams with index value between 0 and (**X**–1) refer to streams that are stored at the second level of the tree network, while those with index between **X** and (**X** + **X**/2 - 1) refer to streams with source nodes at the top level.

We observe that the DC algorithm is fair. All streams suffer the same number of losses independently of their source-node location and the number of hops that they have to traverse to reach destination node. This is imperative for a distributed network, since in this case there is not a unique central server but applications are stored at several places and served by different servers. The DC algorithm guarantees that the quality of video delivered to the users, depends only on the system's capacity and not on the fact that some streams may monopolize the existing resources against the remaining streams. In addition, the placement of the video streams does not affect the quality perceived by the viewers. Usually, the nearer a source is to destination users, the less losses the corresponding streams undergo. However, the DC algorithm achieves the same quality independent of where the video is stored. This can provide the system with flexibility on the placement of the video content.

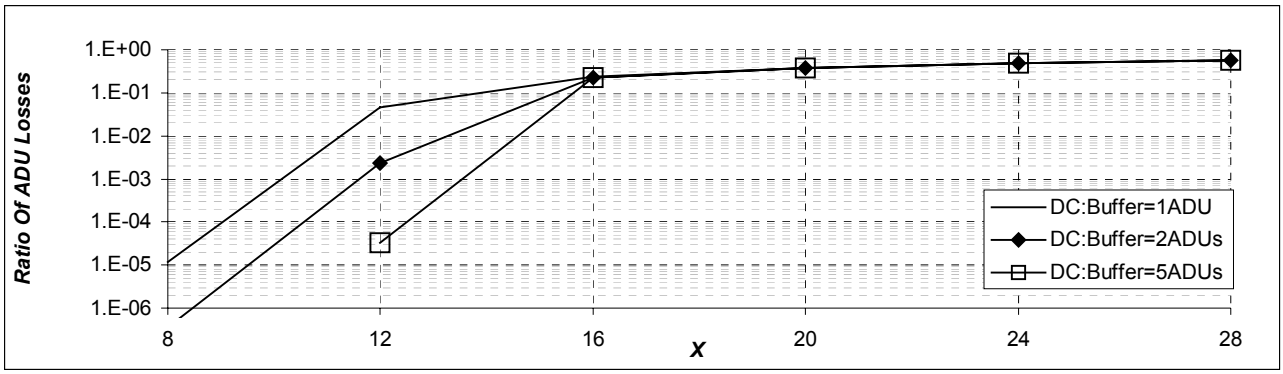

Figure 3.1.2.a: *Ratio of ADU losses versus X for the tree-network*

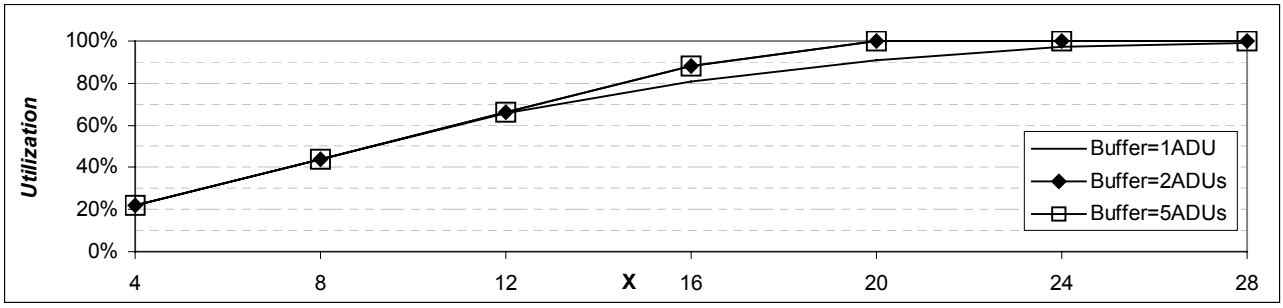

Figure 3.1.2.b: *Utilization of the 1st level (Root Node)*

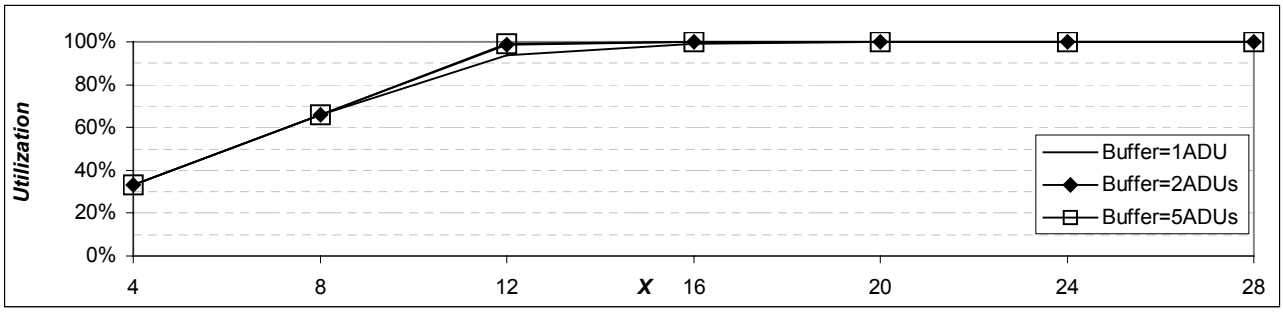

Figure 3.1.2.c: *Utilization of 2nd level (2 Nodes)*

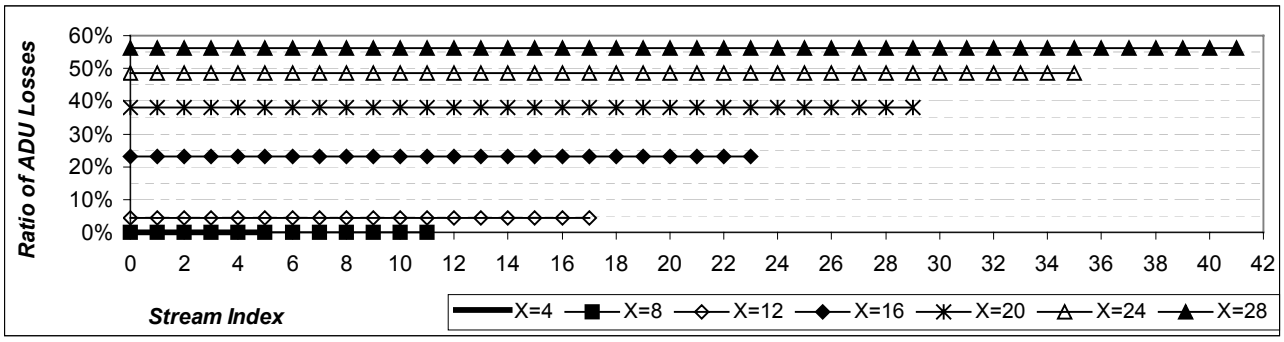

Figure 3.1.2.d: *Loss Distribution across multiplexed streams*

3.2 Scheduling Streams with different frame rates

In modern video-on-demand networks a diverse class of video traffic streams with different requirements is integrated. The nature of video information sources, which varies widely depending on the content of the video, the encoding scheme, and the quality required, outlines the stream's behavior and requirements. There is a wide variation in the volume of information generated when

encoding different signals. Consider for instance two extremes types of video signals: Teleconference video images, on one hand, are nearly always images of people talking while directly facing the camera, usually viewed only from the waist up, and scene changes occur only rarely. These applications have uniform-activity-level; the change in the information content of consecutive frames is not significant. The background usually remains the same and only information about the main objects (typically the faces) needs to be transmitted. With broadcast media, on the other hand, there are no limitations on the content, and scene changes can be frequent. Broadcast television, unlike teleconference video, does not restrict its attention to scenes of talking people. Quite the opposite, there are videos with sudden scene alteration and distinct changes in the subsequent scenes[3]. A great amount of information needs to be transmitted as the consecutive frames differ in content. Uniform-activity-level applications can accomplish an acceptable quality even through a low frame rate. Since this kind of video usually involves low bit rate, it is typically served with a frame rate equal to 10 or 15 frames/second. Broadcast television, on the contrary, requires higher bit rate transmissions for the same quality. On average, it necessitates frame rates equal to 25 or 30 frames/second. As frame rate determines the period of an application, the incorporation of applications with various periods is frequent. In this section we examine the case when the multiplexed streams have different requirements relatively to their frame rate. We investigate the fairness of the DC algorithm for applications with different periods on our distributed video-on-demand system.

3.2.1 Experiment

All the traffic streams are assumed to have uniform length distribution in the interval [1,5]. There are **X** prerecorded streams stored at each node. We assume that a portion of 1/3 of them has period T=50 slots, another 1/3 has period T=100 slots and the remaining 1/3 has period T=200 slots. We simulated the system for **X**=12, 24, 36, 48 and buffer=1, 2 and 5 ADUs for each stream at each node. Each time the same number of streams with different periods reaches destination nodes. That selection was made so that all nodes are evenly loaded. The simulation ends when 300,000 ADUs from the stream with the highest period (T=200 slots) are required at destination nodes. As a result, 600,000 ADUs are required from the streams with period T=100slots and 1,200,000 ADUs from the streams with T=50 slots.

The distribution of losses across video streams as observed from the second level of the tree network is shown in Figure 3.2.1. The y-axis denotes the ratio of ADU losses and the x-axis the stream index. When stream index mod 3 is equal to zero the stream has a period T=50 slots, if it is

[3] Action movies, Formula 1 racing events and football matches are some examples of non-uniform activity-level videos.

equal to one we have a stream with T=100 slots, and, finally, if it is equal to 2 it corresponds to a stream with T=200 slots.

The results are rather disappointing as far as fairness is concerned. Simulation demonstrated that the DC algorithm tries to distribute the number of losses uniformly across streams. However, as the periods of the streams differ, the streams suffer a different ratio of losses. Particularly for large X, the streams with long periods undergo much more losses than the other streams. As a consequence, the streams with short periods monopolize the available bandwidth. Notice that for X=36 or 48 the streams with T=200 slots become absolutely blocked and none of their ADUs is transmitted.

The philosophy of the DC algorithm, as stated in [1], is to schedule from streams the one that has the smaller delay tolerance for the ADU at the head of the queue to expire. This favors streams with small period. For these streams the L_CR counters are updated faster when the network is overutilized; therefore they are selected more frequently. The streams with higher periods are neglected, especially in cases when a large number of streams are multiplexed and many ADU losses occur. A lost ADU is taken into account in the update of the P_CR counter independently of the stream's period; however, an elapsed slot acts in favor of streams with small period compared to streams with higher period. The losses counters of the streams with small period are updated more frequently than those of the remaining streams. At times of network's over-utilization, this acts against the streams with higher periods, which are seldom selected and their loss counters are infrequently updated.

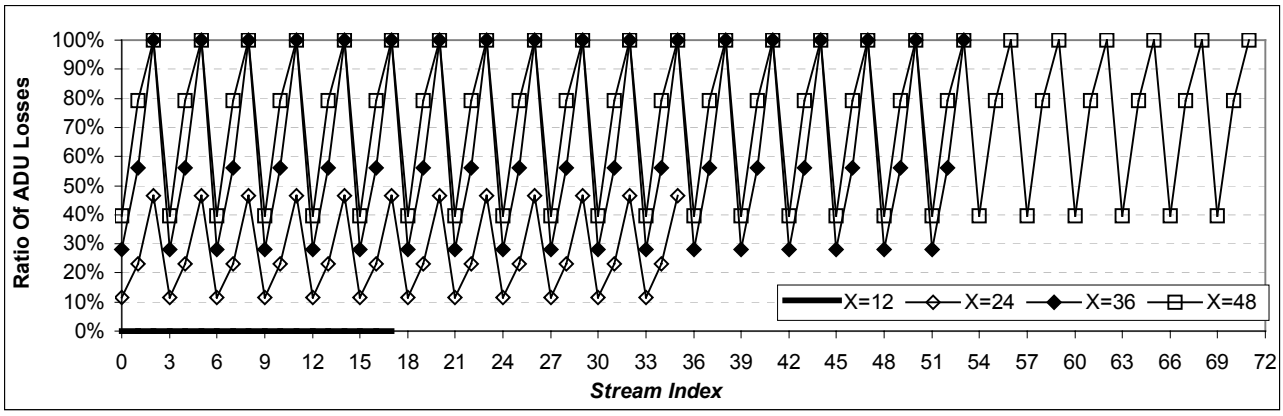

Figure 3.2.1: *Loss Distribution across streams with different periods*

3.2.2 A proposed modification of the DC algorithm

The video quality that the end user perceives depends on the application's period. Even if we achieve to distribute the number of ADU losses uniformly among streams, we would have to attain an equal ratio of losses among streams in order to accomplish the same video quality. For a stream with long period, losing an ADU corresponds to a greater timing interval that the video is degraded compared to losing an ADU from a stream with a smaller period. As a result, if the goal is that all the streams independently of their periods achieve the same video quality, the DC algorithm should

be modified. The goal should be that all streams experience an equal portion of ADU transmissions as well as an equal portion of ADU losses. Ideally, we would like that for each stream *j* the relations: $\frac{L\_CR_n^{j-1}}{L\_CR_n^j} = \frac{T^j}{T^{j-1}}$ and $\frac{data_n^{j-1}}{data_n^j} == \frac{T^j}{T^{j-1}}$ hold, where $data_n^j$ corresponds to the number of ADUs that were transmitted by node *n* from stream *j*. Accordingly, if instead of the relation $P\_CR_n^j = \frac{D\_CR_n^j - T^j * L\_CR_n^j}{T^j}$ (as in [1]), we use the relation $\mathbf{P\_CR}_n^j = \mathbf{D\_CR}_n^j + \mathbf{data}_n^j * \mathbf{T}^j - \mathbf{L\_CR}_n^j * \mathbf{T}^j$ (3.2.2), then among streams with the same data and L_CR values the one with the smallest deadline allowance would be selected. Since $D\_CR_n^j$ counters are increased T^j units for every transmission or loss [1], the streams with small periods will be served first because for the same number of transmissions (data) and losses (L_CR), they have smaller D_CR value. After these streams are served, their data and L_CR counters will be updated, thus in the next examination slot a stream with long period will be selected. Among streams with different data values but equal D_CR and L_CR values, the stream with the least transmissions will be chosen. Likewise, among streams with the same data and D_CR values, the one with the most losses will be chosen to transmit next.

Equation 3.2.2 attempts to balance the number of losses and transmissions among streams. It works in accordance to the relation in [1], when streams have the same period, since for the same losses and transmissions, D_CR plays the primary role. Additionally, equation 3.2.2 eliminates the problem illustrated in the previous section as each time a transmission occurs $data_n^j$ is increased by 1. The selection of a stream now depends not only on the losses it has suffered but also on the total number of transmitted ADUs. An elapsed slot plays the same role independently of the stream's period, while a lost or transmitted ADU influences the priority counter in proportion to the streams period.

Generally, equation 3.2.2 expresses the total time that a stream is ahead or behind. If P_CR > 0 then more ADUs have been transmitted than dropped. Instead, if P_CR < 0, then more ADUs have been dropped than transmitted or the allowable deadline is such that in the next examination slot one or more ADUs will be dropped so that the total number of dropped ADUs exceeds the number of ADU transmissions. Expanding the relation (3.2.2), the term $data_n^j * T^j$ expresses the time that the end user receives faultless video, while the term $L\_CR_n^j * T^j$ expresses the time that the end user experiences video quality degradation. The $D\_CR_n^j$ term is used so that among streams with the same transmissions and losses, the one with the shortest time till expiration is selected. Generally, the relation (3.2.2) attempts to provide all the streams with the same video quality in a time-oriented

manner. The main idea is that fairness will be achieved if all the users receive the same times of faultless and degraded video, respectively.

3.2.3 Simulation Results

The results we obtained from the simulation of the modified DC algorithm are shown in Figure 3.2.3. We simulated exactly the same experiment with the one in 3.2.1 and obtained precisely the same total number of ADU losses. This is expected, since the total number of ADU losses depends exclusively on the load of the system. The difference arises on the dispersal of losses among the contending streams. Notice the absolute uniform distribution of the ratio of ADU losses. All streams independently of their period undergo the same ratio of ADU losses, thus all viewers observe the same video quality. The modification proposed in 3.2.2, is in agreement with the goal that all contending streams obtain the same treatment. No stream dominates or monopolizes the available network resources, instead fair sharing of network's bandwidth has been accomplished among a diverse class of video streams with different timing constraints.

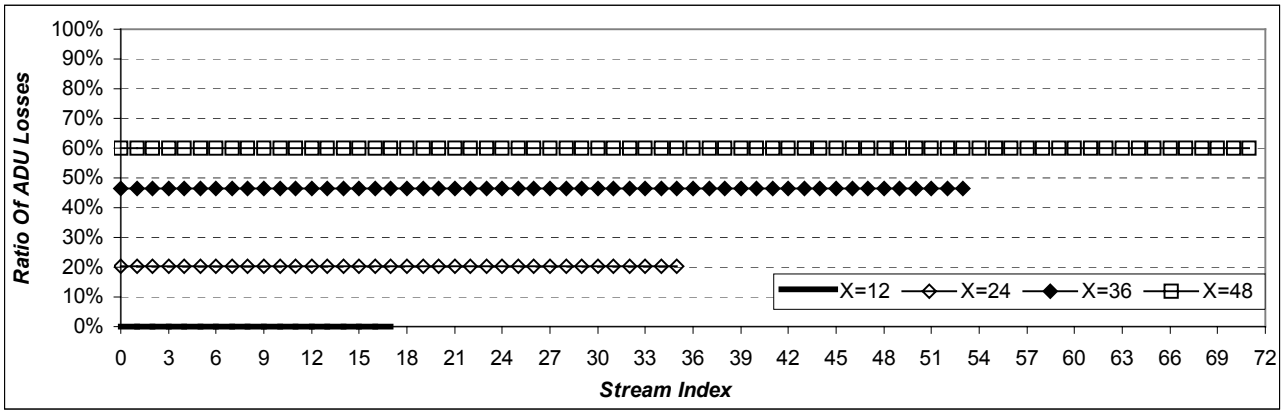

Figure 3.2.3: *Loss distribution across streams with different periods for the modified DC algorithm*

3.3 VCR-like functions

Video on demand (VoD) has the potential of giving individual TV viewers access to a wide range of recorded movies, video programs, games, information and other services. It is distinguishable from more conventional TV viewing due to a high degree of interactivity between the viewer and the material being viewed. Some possible operations that users may wish to perform are VCR-like, e.g. pause, fast forward (FF), rewind (REW), reset, stop etc.

In this section we expand the capabilities of the system to support such interactive users' operations. We adopt the same model for users' behavior as in [2] which enables pause, fast forward, rewind and play operations. We simulate an experiment similar to the one in section 3.1.

3.3.1 Experiment

Each user can be either in PLAYBACK state or in VCR state. PLAYBACK state represents the sequential access of a video where the frames travel regularly from source to destination. VCR state stands for users that perform fast forward, rewind or pause operation. When a user performs a pause function, the traffic flow stops and the video that the user receives remains unchanged. For FF or REW operations the video program is displayed ***n*** times faster than the normal display rate forward or backward, respectively. We refer to the multiplier ***n*** as the *VCR Factor*[4].

In this work, we approach the problem of supporting VCR operations by storing at source nodes a separate version of each program which although is assumed to be encoded at normal rate (e.g. 30 frames/second), it does not contain all the initial frames. Several intermediate frames have been skipped and only the most significant frames are stored. The latter frames could be for example those ADUs that correspond to a video scene change and contain significant information volume. For instance, these can be the frames that in an MPEG format encoding correspond to I frames or I and P frames of the video stream[5]. The characteristic of this version of the video is that the ADUs have longer average length. Since only the most significant frames of the video are stored at this version, these ADUs will be of the maximum length. Consequently, when a stream is in VCR state, its average cell generation rate increases. When a FF or REW operation is performed, the frames of this version start flowing, giving the user the sense that the program speeds up. In reality, the rate that frames are displayed does not change but the absence of the intermediate frames gives the feeling that the program speeds up.

All the video streams are again supposed to be characterized by a uniform ADU length distribution in the interval [1,10] and a period equal to $T^j = 100$ slots. Furthermore, we assume that a second version of each video is available at each video's source node for the FF and REW operations. As mentioned before, this version contains only the most significant ADUs. The ADUs of the second version are assumed to have a uniform length distribution in the interval [9,10] and a display period equal to $T^j = 100$ slots.

3.3.2 Simulation Results

In the case of the VCR model, too, simulation showed that for large **X** values, when the system is overutilized, the curves for buffer=1,2,5 ADUs converge, since the buffer cannot be exploited. On the other hand, for small **X** values, we noticed a slight improvement as available buffer space

[4] Common values for the VCR factor are between 2 and 6.

[5] Consider for example MPEG encoding format with a frame rate equal to 30 frames/second and a 6 frames group of pictures. If only I frames are stored in the VCR version, then the program will appear as speeding up 6 times, hence the corresponding VCR factor=6.

increases because the DC exploits the available bandwidth and buffer space to send ADUs in advance.

In figure 3.3.2.b, we give the distribution of ADU losses across streams. Generally, we observe that the DC algorithm treats fairly the multiplexed streams when the VCR-like operations are supported.

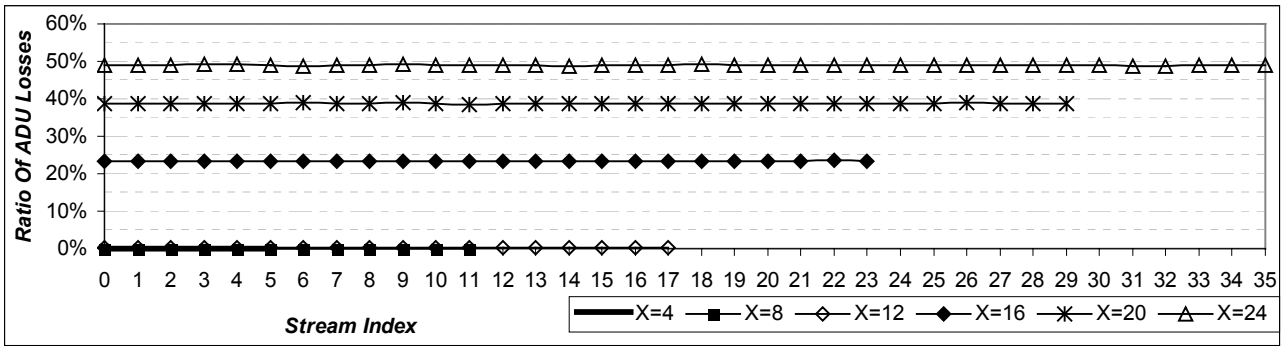

Figure 3.3.2.b: *Loss Distribution across multiplexed streams for the VCR experiment*

4 Performance evaluation of DC Algorithm for MPEG-4, H.263 video traces

In real videos, there are some dependencies between the consecutive frames as well as a non-uniform frame length distribution. The scope of the following experiments is to extend the conclusions drawn from the results of the previous section to the case of actual video traces.

4.1 MPEG-4 videos

In the following experiments we use MPEG-4 video trace files originating from existing movies and other video programs. The video programs we have chosen to use in our experiments cover a wide range of video applications such as action movies, cartoons, tele-conference, ski, Formula 1 racing events, music video clips, talk shows, etc.

In the MPEG-4 encoding videos we have used, the number of video objects is set to one, i.e., the entire scene is a single object. The width of the display is 176 pels, the height 144 pels and the depth is 8 bits per pel. The rate at which video frames are generated was chosen to be 25 frames/sec. Each video is encoded at three different quality levels: high, medium and low quality. For the low quality encoding the quantization parameters were fixed at 10 for I frames, 14 for P frames and 18 for B frames. For the medium quality encoding the quantization parameters for all three frame types were fixed at 10, whereas for the high quality encoding at 4.

The Group of Pictures (GoP) pattern was set to I B B P B B P B B P B B. I frames are independently encoded using the Discrete Cosine Transform (DCT). The DCT coefficients are then quantized and variable-length-coded. P frames are forward predicted from the preceding I or P frames using motion vectors. Finally, B frames are predicted from the preceding I (or P) frames and

the succeeding P (or I) frames. The predictions errors are DCT encoded, quantized, and variable-length-coded[6].

Due to these dependencies not all the frames have the same significance relatively to the image that the end viewer actually perceives. I frames are vital for all the subsequent frames of the group to be displayed, so if an I-type frame is lost, then the whole group is considered to be lost and none of the remaining frames is transmitted. Additionally, if a P-type frame is lost, the rest of the group is considered to be lost and the subsequent frames are not transmitted. Finally, a B-type lost frame accounts for a single loss, since no frame depends on B frames.

4.2 Ratio of ADU losses versus bandwidth

In Figure 4.2, we observe the ratio of ADU losses versus the bandwidth[7] for **X**=12 as taken from simulation experiments after an hour of network operation when high quality video programs were used. Notice that for small values of the available bandwidth (smaller than 9.54 Mbps), the DC algorithm performs rather independently of the buffer space. This indicates that the system is highly loaded and the buffer space cannot be exploited. In contrast, as the available bandwidth increases, buffer space affects the performance of the algorithm. Particularly, for bandwidth equal to 22.26 Mbps, the case of buffer=1 ADU gives almost 2702 times more losses compared to the cases with buffer=2 or 5 ADUs. This demonstrates the strong influence of the available buffer space on the performance of the DC algorithm. When the system is lightly loaded the DC exploits the available bandwidth and buffer space to forward ADUs at down stream nodes and reduce the ADU losses.

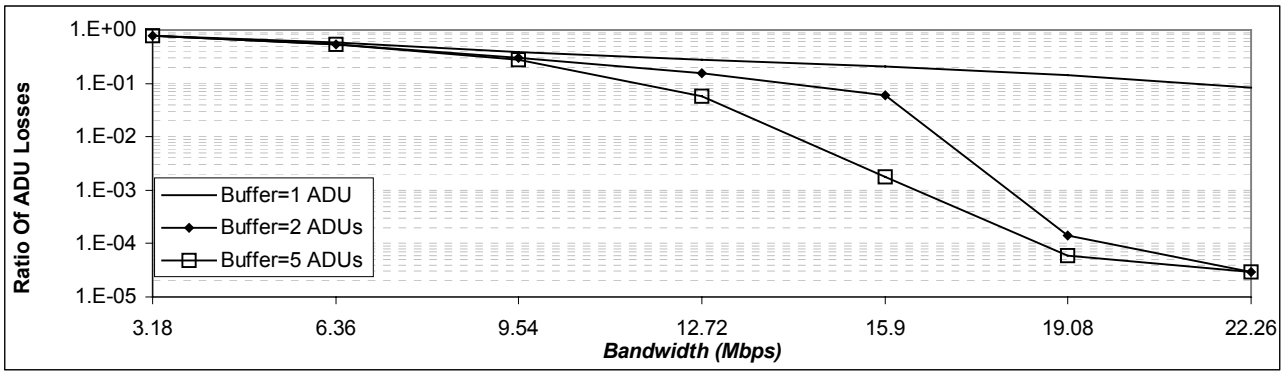

Figure 4.2: *Ratio of ADU losses for Buffer=1,2 and 5 ADUs versus available bandwidth*

[6] A detailed description of the coding algorithm can be found in [15].

[7] The term bandwidth actually refers to the maximum output rate at each headend. The values of the x-axis in Figure 4.2 are calculated as follows: Let suppose that each video stream has period T slots. This means that one frame has to be available at the receiving end every T slots, and accordingly, 25 frames have to be available at the receiving end within 25*T slots. We refer to video with 25 frames/sec, i.e. 25*T slots correspond to 1 second. During a slot 424 bits are transmitted, consequently the "bandwidth" is equal to 25*T*424 bits/sec. For example, for T=300 slots, "bandwidth"=3.18 Mbps (i.e. the first value in the x-axis). The actual total bandwidth for the system in each case is 4 times the corresponding value in the x-axis.

4.3 Experiment with high, medium and low quality MPEG-4 videos

We simulate approximately one hour of operation for the binary tree network for **X**=4, 8, 12, 16, 20, 24 and 28 programs, and buffer=1, 2 and 5 ADUs of the maximum length. We present the simulation results for a bandwidth of **12.72 Mbps** and for high, medium and low quality video.

4.3.1 Ratio of ADU Losses

As far as the ratio of ADU losses is concerned, we notice important differences between different qualities of video programs. Generally speaking, we observe that the curves that correspond to high quality vary significantly from those that correspond to medium or low quality. Comparing the medium and low quality loss curves, we observe that they have almost the same form with medium quality giving higher ratio of ADU losses.

Furthermore, we notice that the buffer plays an important role in the ratio of ADU losses, especially in the cases of medium and low quality videos. For high quality video, on the other hand, buffer plays a significant role only when the number of the multiplexed video streams is small (X<16). When more streams are multiplexed, the network becomes overloaded and almost no ADUs can be sent in advance, thus the buffer space is not exploited.

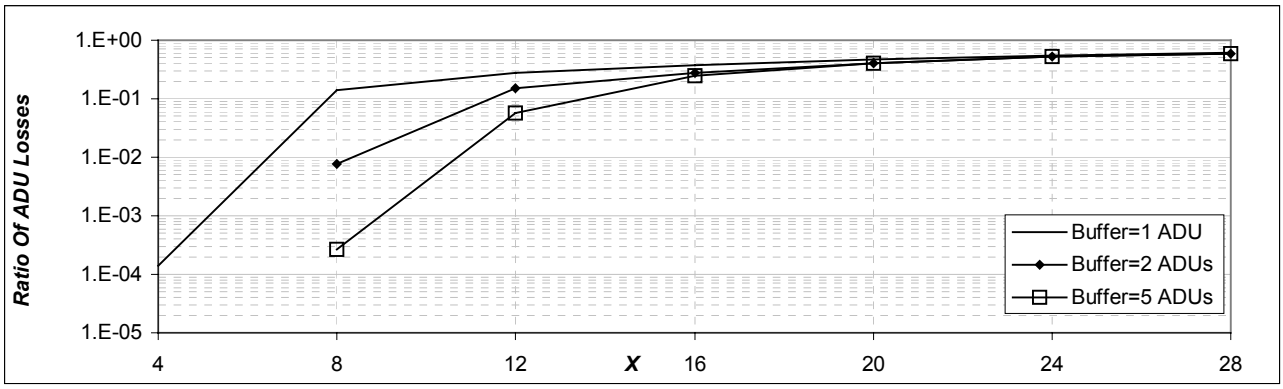

Figure 4.2.1.a: *Ratio of ADU losses versus X for high quality video*

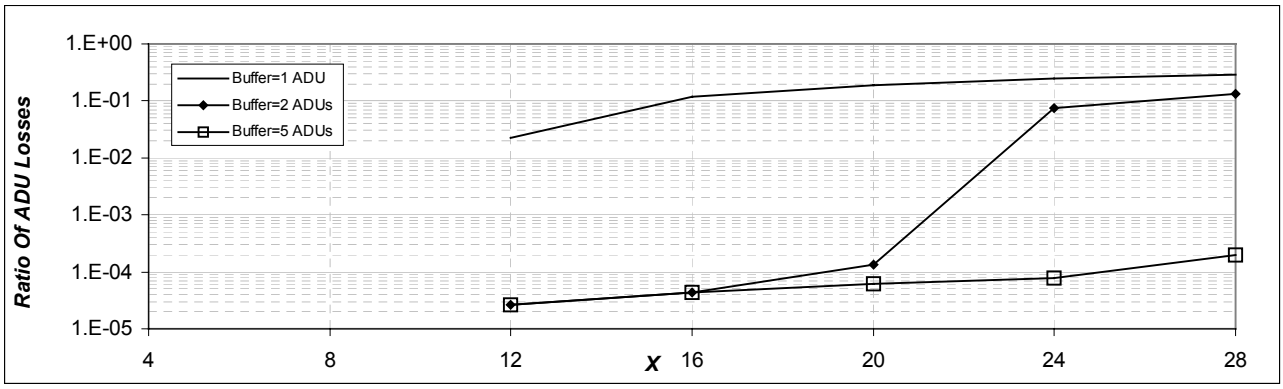

Figure 4.2.1.b: *Ratio of ADU losses versus X for medium quality video*

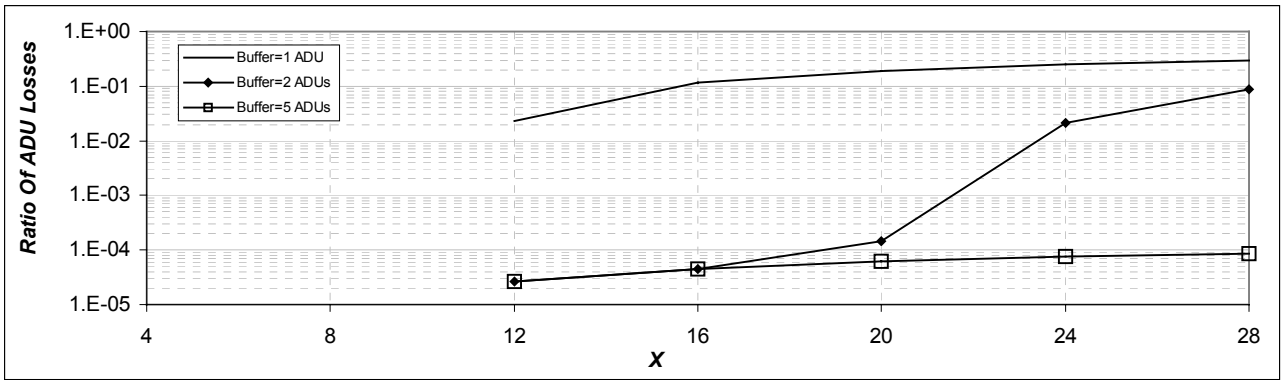

Figure 4.2.1.c: *Ratio of ADU losses versus X for low quality video*

4.3.2 Nodes' Utilization

In case of high video quality programs, we notice that the utilization of the nodes at the second level is higher when comparing it to the cases of medium and low video quality programs. We also notice that as the buffer size increases so does the node utilization. This is indicative of the fact that buffer affects the performance of DC algorithm, since the available buffer is used to forward ADUs, reduce losses and increase utilization. However, when the system becomes overloaded and several losses occur, buffer cannot be utilized to reduce losses.

In case of medium and low quality video programs, we also observe that as **X** and buffer increases, so do the utilizations. Notice that in both cases the three curves (Buffer=1,2,5) separate at the X values where the losses differ (see Figures 4.2.1.b, 4.2.1c). As explained earlier, this is because the buffer space is exploited to send ADUs in advance and consequently the utilization increases.

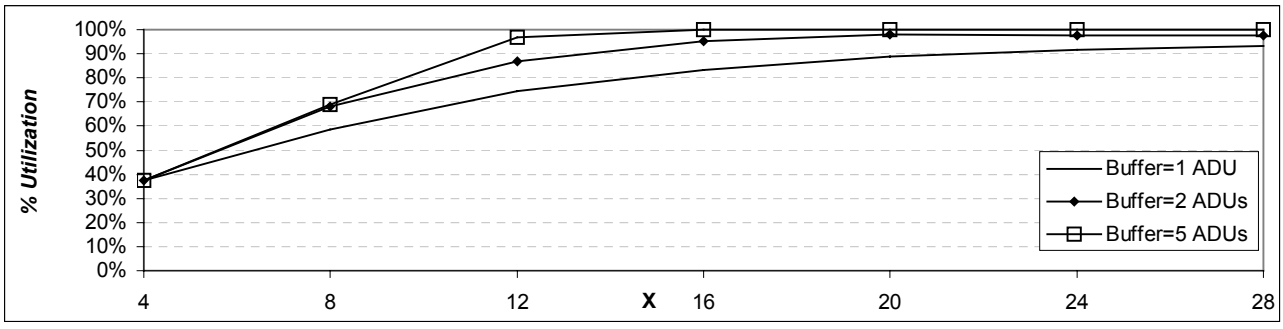

Figure 4.2.2.a: *% Utilization of the nodes of second level of the tree– High Quality*

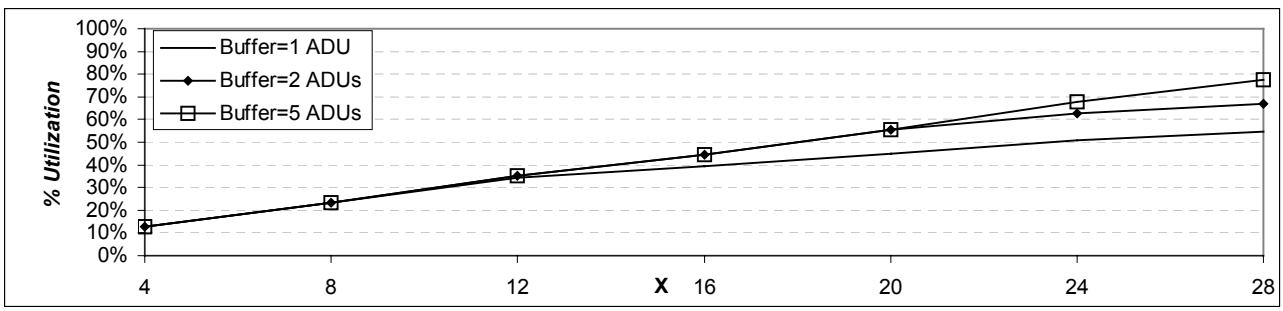

Figure 4.2.2.b: *% Utilization of the nodes of second level of the tree– Medium Quality*

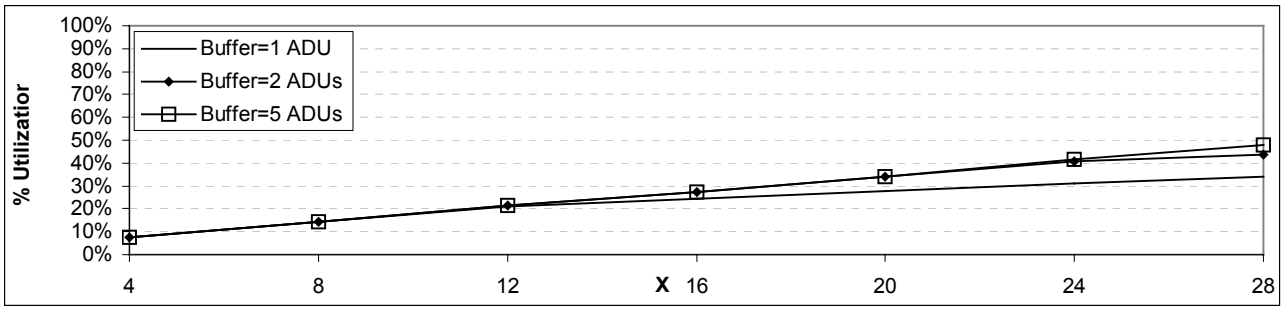

Figure 4.2.2.c: *% Utilization of the nodes of second level of the tree– Low Quality*

4.3.3 Distribution of ADU losses across streams

Our simulations have shown that in each of the three cases examined, the DC algorithm distributes fairly the losses across the multiplexed video streams. These results not shown here due to lack of space demonstrate the fact that DC works well for actual video files, too. Therefore, we can claim that the DC algorithm operates fairly for every type of MPEG or other encoding method. The format of the GoP does not actually affect the performance of the algorithm as long as the frame rate is constant.

Another interesting conclusion that can be drawn is that the placement of the video programs can be chosen independently of the video programs' popularity. As stated in section 2.2, the video program's popularity determines its placement in order that the most popular programs are stored closer to the users, whereas the least popular ones are stored at the upper levels of the network tree topology. This choice is made so that the most popular videos are located closer to the users, and suffer the least start up delay and the least number of frame losses. However, our experiments have shown that all streams independently of the level at which they are stored suffer the same number of losses. Based on this observation, we can claim that DC algorithm also gives flexibility to the system configuration regarding the placement of the video programs. The video programs can be placed at any level of the network hierarchy without their quality relative to frame losses being affected.

In our experiments we have assumed that each program is required only by one children stream node. In a real system, some videos may be required by both of a node's children stream nodes, while others may not be active. As revealed by our experiment, the way the DC algorithm distributes the losses across the multiplexed streams depends exclusively on the number of multiplexed streams and the available bandwidth. Since the DC algorithm distributes the losses fairly across the multiplexed video streams independently of the videos' contents, we can assume that the DC algorithm is also fair when some videos are required by more than one headends.

4.3.4 Ratio of I-, P-, B-frame losses

Our simulation results also demonstrate that the ratios of I-, P- and B-frame losses are almost the same for all the multiplexing streams. In Figure 4.2.4, we present the simulation results for buffer=1 ADU and X=12 streams when high quality video programs are used. The streams in the x-axis correspond to the streams that are prerecorded at the first and the second level of the tree. Notice the agreement in the ratios of I-, P- and B-frame losses among the multiplexed video streams. This is another indication of the fairness of the DC algorithm. Since the same number of I-, P- and B-frame losses occurs among different streams irrespective of the tree level at which they are stored, the users at the receiving ends obtain almost the same video quality.

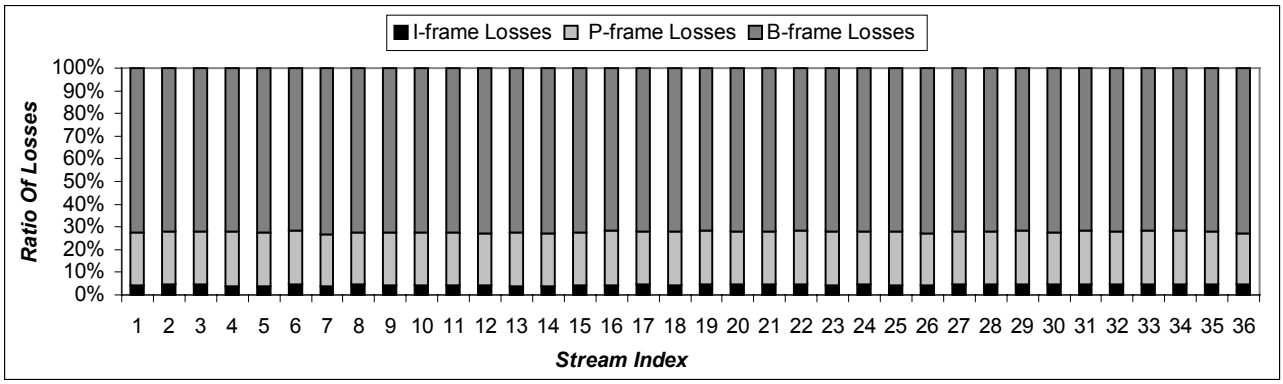

Figure 4.2.4: *Ratio of I-, P-, B-frame losses for Buffer=1 ADU, X=12*

4.4 H.263 videos

In the following experiment we have used H.263 video compression with variable bit rate, too. The width and the height of the display is the same as in the MPEG-4 format, i.e. 176 and 144 pels, respectively. The rate at which video object planes are generated was chosen to be 25 frames/sec, i.e. 40 msec.

We enable PB-frames, also called "stuffed" frames, which have a period equal to 80 msec. A PB-frame consists of two consecutive frames that are encoded as one unit. Specifically, a PB-frame consists of a P-frame, which is predicted from the preceding P-frame, and a B-frame, which is bi-directionally predicted from the preceding P-frame and the P-frame being part of the PB entity. When the reconstruction of the PB-frame is complete, the B-frame is displayed first for a period of 40 msecs and then the P-frame for also a period of 40 msecs[8].

H.263 encoding produces I (usually only the first frame is an I-fame), P and PB frames. I and P frames have a period equal to 40msecs, whereas PB frames have a period equal to 80msecs. Consequently, H.263 may produce variable frame periods. Videos with plenty of information generate, except of PB-frames, many I- and P-frames, thus they have often period-alterations between 40 and 80 msecs arising from the frames generated. On the contrary, videos with low scene

[8] A detailed description of the coding algorithm can be found in [15].

alteration and uniform level activity (such as Lecture Room Camera, Office Camera etc) generate almost exclusively PB-frames, leading to an unchanged period equal to 80 msecs.

4.5 Experiment with MPEG-4 and H.263 videos

The purpose of this experiment is to test and further support the modification we proposed in section 3.2.2 relatively to the fairness of the DC algorithm when video applications with different frame rates are supported in the same network. For this reason, we have chosen that each time half of the video programs are MPEG-4 encoded (high, medium and low quality), whereas the other half of the programs are H.263 encoded. When an H.263 program contains ample information and many scene alterations, sport events for example, it does not contain exclusively PB frames, but I and P frames as well. However, for the purpose of our experiment, we have chosen only programs with a few scenes alterations (e.g. video conference programs) that contain information, which can be presented exclusively with PB frames. Consequently, from the MPEG-4 encoding video programs the transmission of one frame per 40 msecs will be required, while from the H.263 encoding video programs the transmission of one "stuffed" frame per 80 msecs will be required.

4.5.1 Experiment Description

At each node they are stored **X** prerecorded video streams that supply the children nodes with traffic. **X/2** streams are high, medium and low quality MPEG-4 encoded and the remaining **X/2** are H.263 encoded. The available bandwidth is assumed equal to **9.54 Mbps**.

In case a frame is dropped, if it belongs to an MPEG-4 program, we follow the same process as described previously. If the frame belongs to an H.263 video program, we count it for loss but we transmit it even if its allowable delay has expired, due to the fact that the subsequent frames depend on it.

We have considered two cases. In the first case, we use the relation for the priority index counter as stated by the authors of [1]. In the second case, we use the modified relation for the P_CR we proposed in section 3.2.2. In these two cases, we have used the same placement of video programs in order that the video placement does not affect the results. We are interested in the way the losses are distributed across video streams with different frame rates.

4.5.2 Simulation Results and Discussion

Figure 4.5.2.a shows the simulation results when the P_CR is defined as in [1]. Streams with odd index numbers refer to MPEG-4 video programs, while streams with even index numbers refer to H.263 video programs. Notice that the ratio of ADU losses depends on the period of the stream. Specifically, we observe that the ratio of ADU losses for the H.263 encoding video streams is 50 % higher than the corresponding ratio for MPEG-4 encoded video programs. This can be ascribed to

the fact that H.263 programs have double the period of MPEG-4 programs and that the DC algorithm as stated in [1] distributes evenly the number of ADU losses across streams independently of their periods. However, this does not give equal ratios of ADU losses when the multiplexed streams have different periods.

In Figure 4.5.2.b we present the simulation results for the modified DC algorithm. Notice the uniform distribution of the losses across multiplexed streams. All the video streams, independently of their period, undergo almost the same ratio of ADU losses. This uniformity proves that the proposed modification for the priority index counter works well for real video streams with different frame rates too. This is important in real systems because it shows that video streams with different frame rates and different encodings can be multiplexed with a fair bandwidth allocation. Several video applications can be multiplexed and only the available bandwidth and the number of multiplexed streams affect their image quality. The rate at which their frames are generated does not affect the ratio of frame losses they undergo.

In our experiment, we have assumed that each time half of the programs are MPEG-4 encoded and the remaining half are H.263 encoded. However, since we observe the same ratios of ADU losses independently of the encoding method, we can assume that the DC algorithm is fair for every mixture of MPEG-4 and H.263 video programs.

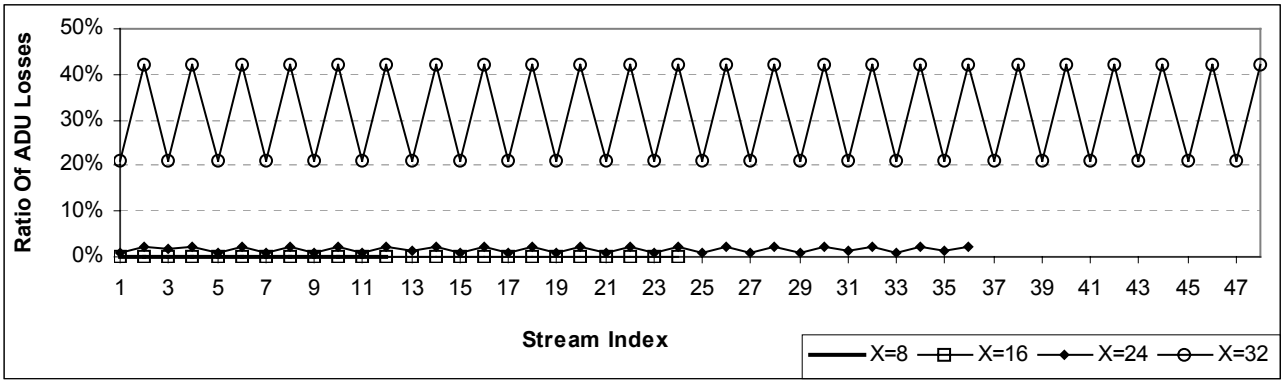

Figure 4.5.2.a: *Loss Distribution across MPEG4 and H.263 video programs according to [1]*

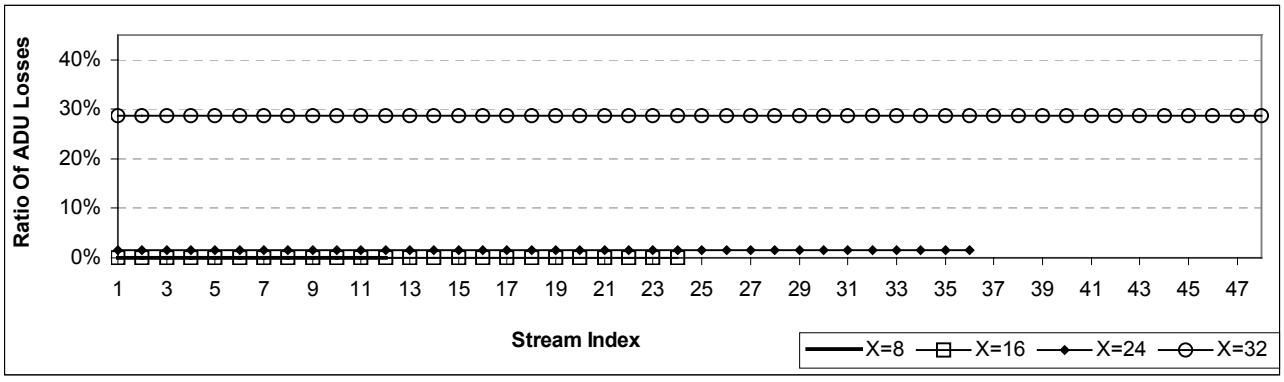

Figure 4.5.2.b: *Loss Distribution across MPEG4 and H.263 video programs for the modified DC algorithm*

5. Conclusions

In this work, we examined via extensive simulations the performance of an innovative scheduling approach for prerecorded soft real-time applications, referred to as the Deadline Credit Algorithm. The DC algorithm uses dynamic allocation of bandwidth and distributes the available bandwidth fairly between all the competing video streams. We found that when the system is under-utilized, the DC algorithm forwards an equal number of ADUs from each traffic flow at down stream nodes and keeps the streams at the same level of delay. On the other hand, when the network is over-utilized and frames need to be dropped, DC distributes the losses fairly across streams. The efficiency of the DC algorithm with regard to the ADU losses increases as the available buffer space at the traversing nodes increases, because the DC algorithm exploits the available buffer space to send ADUs in advance. Furthermore, when the hops of a network are unequally loaded, the DC algorithm builds deadline credit dynamically by utilizing the available bandwidth and buffer space, and remains fair among multiplexed streams.

We evaluated the performance of the DC algorithm for a distributed Video-on-Demand network, which fits the existing tree topology used in today's cable TV systems. Simulation results showed that the DC algorithm is fair among the multiplexed video streams. This means that all streams suffer the same number of losses independently of their source-node location. Accordingly, the placement of the video streams does not affect the quality perceived by the viewers, and the DC algorithm achieves the same quality independent of where in the network the video is stored. However, in this case too, the ratio of ADU losses depends on the available buffer space. The latter is used to store the ADUs that are forwarded by the DC algorithm.

The only case where the DC algorithm was unfair was when a number of streams with different frame rates were multiplexed. Simulations demonstrated that the DC algorithm tries to distribute the number of losses uniformly across streams. However, as the periods of the streams differ, the streams suffer a different ratio of ADU losses. The modified DC algorithm we proposed works fairly in this case, too. Particularly, with the modified DC algorithm, all streams independently of their period undergo the same ratio of ADU losses, thus all viewers observe the same video quality. Moreover, we simulated the network when VCR-like functions are supported and we then also observed that the DC algorithm treats fairly the multiplexed streams.

Finally, we extended the validity of our results by simulating actual videos encoded in MPEG-4 and H.263 format. In these cases, as well, the DC algorithm is fair and distributes losses evenly across multiplexed streams. This can contribute to a flexible placement of the video programs, since the placement can be chosen independently of the video programs' popularity. Finally, we provided simulation results that support the modification we proposed. The conclusion is that the modified DC algorithm treats fairly videos with different frame rates, with different quality encodings and

with different encoding methods. This means that a diverse class of video applications can be integrated into the same network and all these applications are treated fairly.

References

[1] Z. Antoniou, I. Stavrakakis, "Efficient End-to-End Transport of Soft Real-Time Applications", pp. 470-482, IFIP Networking'2000, May 14-19, 2000, Paris, France

[2] Constantinos C. Vassilakis, "Modeling, Design and Performance Evaluation of Interactive Distributed Video-On-Demand Systems", M.Sc. Thesis, ECE Department, Technical University of Crete, 1999. Part of this M.Sc. Thesis has been published in the ACM Multimedia Systems Journal, Vol. 8, No. 2, 2000, pp 92-104, under the title "Video Placement and Configuration of Distributed Video Servers on Cable TV Networks", (authors: C. Vassilakis, M. Paterakis and P. Triantafillou).

[3] Z. Antoniou, I. Stavrakakis, "Deadline Credit Scheduling Policy for Prerecorded Sources", IEEE GLOBECOM'99, Dec. 5-9, 1999, Rio, Brazil.

[4] R. Onvural,"Asynchronous Transfer Mode Networks - Performance Issues", Artech House, 1995

[5] UYLESS D. BLACK, (Second Edition), "Data Communications And Distributed Systems", Prentice-Hall International Editions, 1987

[6] Daniel Minoli, "Video Dialtone Technology", McGraw-Hill, Inc, 1995

[7] Naohitsa Ohta, "Packet Video Modeling and Signal Processing", Artech House, 1994

[8] Rafael C. Gonzalez, Richard E.Woods, "Digital Image Processing", Addison-Wesley Publishing Company,1992

[9] R. J. Clarke, "Digital Compression of still images and video", Academic Press, 1995

[10] Moving Pictures Expert Group Organization, http://www.mpeg.org

[11] P. Cherriman, L. Hanzo and R. Lucas, ARQ-assisted H261 and H263-based Programmable Video Transceivers, 1995/6 Research Journal, Communications Group, University of Southampton, http://www.ecs.soton.ac.uk/publications/rj/1995-1996/comms/pjc94r/rj95.htm

[12] H261 Video Coding, http://www-mobile.ecs.soton.ac.uk /peter/h261 /h261.html

[13] H263 Video Coding, http://www-mobile.ecs.soton.ac.uk/peter/h263/h263.html

[14] H261 Overview, http://www.nyquist-media.co.uk/streaming/h261.html

[15] Frank H.P. Fitzek and Martin Reisslein, "MPEG-4 and H.263 Video Traces for Network Performance Evaluation", Technical University Berlin, Telecommunication Network Group, TKN Technical Report Series, TKN-00-06, October 2000, http://www-tkn.eetu-berlin.de/research/trace/trace.html

[16] Joan Mitchell, William B. Pennebaker, Chad E. Fogg, and Didier J. LeGall, "MPEG Video Compression Standard", International Thomson Publishing, 1997

[17] Jean-Pierre Leduc, Digital Moving Pictures – Coding and Transmission on ATM Networks, Volume 3, Elsevier, 1994

[18] C. Bisdikian and B. V. Patel, "Issues on Movie Allocation in Distributed Video-on-Demand Systems", IEEE International Conf. on Communications, 1995, pp 250-255. Also published in IEEE Multimedia Magazine under the title "Cost-Based Program Allocation for Distributed Multimedia-on-Demand Systems", IEEE Multimedia Magazine, Vol. 3, No. 3, Fall 1996

[19] ATM Forum, http://www.atmforum.com

[20] J.P. Nussbaumer, B. V. Patel, F. Schaffa, and J. P. G. Sterbenz, "Network Requirements for Interactive Video on Demand", IEEE Journal on Selected Areas in Communications, Vol. 13, No 5, June 1995.

[21] G. Bianchi, R. Melen, " Performance and Dimensioning of a Hierarchical Video Storage Network for Interactive Video Servers", European Transactions on Telecommunications, Vol 7, No. 4, July-August 1996