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

Adamantia Alexandraki and Michael Paterakis

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

Abstract. 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 in communication networks 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 QoS, and the fairness of the scheduling mechanism. We examine cases with homogenous and diverse video streams. 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.

1. Introduction

Telecommunication networks have lately extended to support a variety of services to the users such as tele-education, remote working, high-definition TV, web video streaming, etc. The demand of all these applications led to the concept of ***broadband integrated digital networks (B-ISDN)***, which are systems of high capabilities. The role of the ***scheduling process*** in B-ISDNs is to select each time the application that should be served next, so that the efficiency of the network is increased and the Q*uality of Services (QoS)* is kept at a satisfactory level.

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 consecutive 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. In this work we use VBR-encoded video, instead of CBR video, because for the same

E. Gregori et al. (Eds.): NETWORKING 2002, LNCS 2345, pp. 202-213, 2002.

image quality VBR-encoded video can have a significantly lower average bit rate as well as exhibit considerable multiplexing gain.

We use the Asynchronous Transfer Mode (ATM) technique of switching and multiplexing multimedia streams, which encapsulates data 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.

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 for homogenous as well as diverse video streams. 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

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 time 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:

$$\mathrm{P\_CR}_n^j = \frac{D\_CR_n^j - T^j * L\_CR_n^j}{T^j} \quad (1)$$

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 streams.

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.

2. Video-on-Demand System

Distributed information systems are 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 paragraphs 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].

The distribution network topology is considered to be a binary tree associating a number of video servers and switching nodes connected via communication links[2]. At each node $N_{i,j}$ there is a storage device where some prerecorded video streams $X_{i,j,k}$ are stored, and a video server $VS_{i,j}$, 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

[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.

[2] We assume that the tree network is of binary form, although, in practice, the location of the distributed servers may differ.

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.

All the nodes of the tree network except those that belong to the last level run the DC algorithm for the streams they serve. The nodes of the last level, consume the arriving ADUs as well as the ADUs that are prerecorded at these nodes. At each system node there is also 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.

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.

Although there are many options relatively to the number of video programs stored at each node, in this work, 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.

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. 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.

In the following experiments we assume a distributed video-on-demand system of depth three. The results shown below have been computed as average values from at least 10 replicated experiments (Monte Carlo Simulation). In each case, the simulations were of sufficient duration for the system to attain a steady state.

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 (i.e. streams with the same periods). The main conclusion that we drew (results are omitted due to lack of space) is 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. 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.

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. 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. With broadcast media, on the other hand, there are no limitations on the content, and scene changes can be frequent. There are videos with sudden scene alteration and distinct changes in the subsequent scenes[3]. Uniform-activity-level applications can accomplish an acceptable quality even through a low frame rate, thus they are 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, 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.

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

The distribution of losses across video streams as observed from the second level of the tree network is shown in Fig. 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 0 the stream has a period T=50 slots, if it is equal to 1 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

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

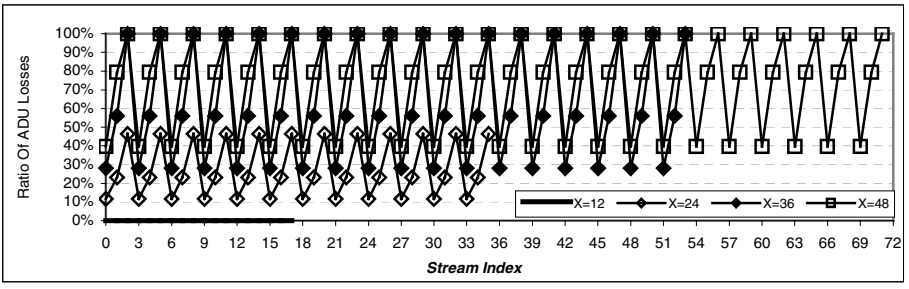

Fig. 1. Loss Distribution across streams with different periods

different ratio of losses. Particularly for large X, the streams with long periods undergo much more losses than the other streams.

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. 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.

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. Let the $data_n^j$ correspond to the number of ADUs that were transmitted by node n from stream j. Then, if instead of the relation (1), as in [1], we use the relation

$$\mathbf{P\_CR}_n^j = \mathbf{D\_CR}_n^j + \mathbf{data}_n^j * \mathbf{T}^j - \mathbf{L\_CR}_n^j * \mathbf{T}^j \quad (\mathbf{2})$$

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 (2) attempts to balance the number of losses and transmissions among streams. It works in accordance to (1), when streams have the same period, since for the same losses and transmissions, D_CR plays the primary role. Additionally, equation (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 (2) expresses the total time that a stream is ahead or behind. Expanding the relation (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 (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 Fig. 2. 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.

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.

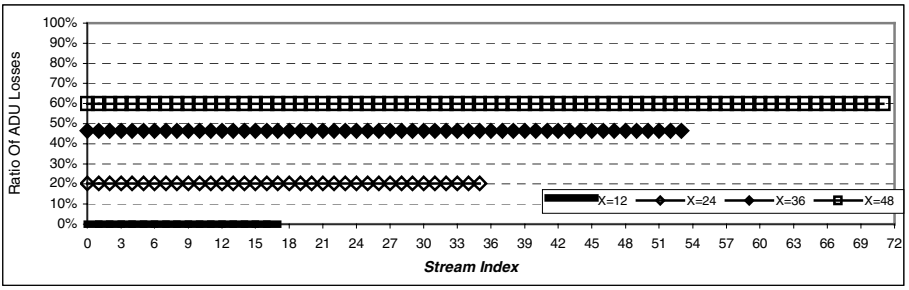

Fig. 2. Loss distribution across streams with different periods for the modified DC algorithm

4.1 MPEG-4 Videos

In the following experiments we use MPEG-4 video trace files originating from existing 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. Each video is encoded at three different quality levels: high, medium and low quality[4].

The Group of Pictures (GoP) pattern was set to I B B P B B P B B P B B. Due to the dependencies between frames 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 Experiment with 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** for high quality video[5].

Our simulations have shown that in each of the three cases examined, the DC algorithm distributes fairly the losses across the multiplexed video streams and gives almost the same ratios of I-, P- and B-frame losses to the multiplexing streams. Notice in Fig. 3 the agreement in the ratios of I-, P- and B-frame losses among the multiplexed video streams (i.e. streams of the first and second level of the tree). This is another indication of the fairness of the DC algorithm.

---

[4] For an extended description of the videos we used, the read is referred to [15].

[5] Simulation results are also available for medium and low quality video, but are omitted due to space constraints.

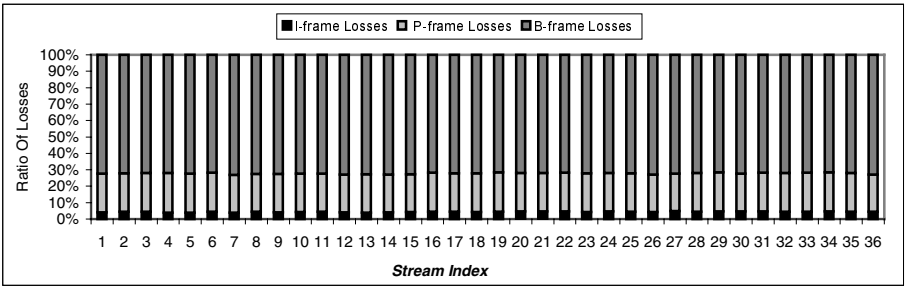

Fig. 3. Ratio of I-, P-, B-frame losses for Buffer=1 ADU, X=12, Bandwidth=12.72 Mbps

Another interesting conclusion that can be drawn is that the placement of the video programs can be chosen independently of the video programs' popularity. What usually happens is that 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 [2],[18]. 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.

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. However, notice that the location of content still has a potential impact on network utilization and user startup latency.

4.3 H.263 Videos

In the following experiment we have used H.263 video compression with variable bit rate, too. We enable PB-frames, also called "stuffed" frames. A PB-frame consists of two consecutive frames that are encoded as one unit [15].

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 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.

4.4 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 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. 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. The available bandwidth is assumed equal to **9.54 Mbps.**

Fig. 4 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. In contrast, for exactly the same experiment, as shown in Fig. 5 the modified DC algorithm gives a uniform distribution of the losses across multiplexed streams.

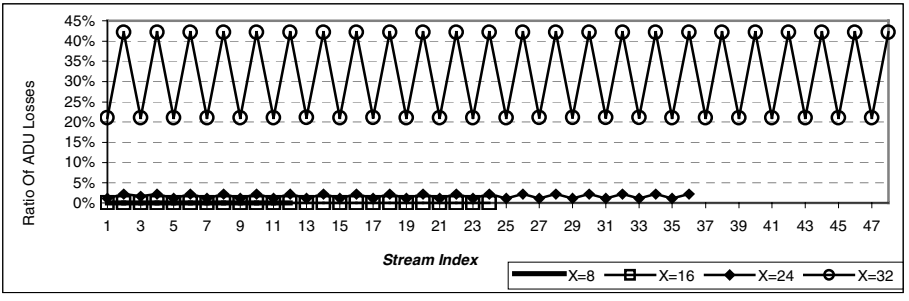

Fig. 4. Loss Distribution across MPEG4 and H.263 video programs according to [1]

5. Conclusions

The DC algorithm uses dynamic allocation of bandwidth and distributes the available bandwidth fairly between all the competing video streams. 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.

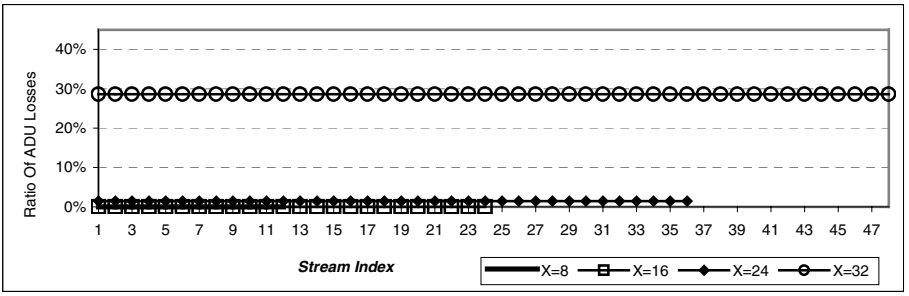

Fig. 5. Loss Distribution across MPEG4 and H.263 video programs for the modified DC algorithm

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, and validate our results by simulating actual videos encoded in MPEG-4 and H.263 format. The main conclusion we drew is that the DC algorithm is fair and distributes losses evenly across multiplexed streams. This means that all streams suffer the same number of losses independently of their source-node location. 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, i.e. 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.ee.tu-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