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 Electronic and Computer Engineering, Laboratory of Information and Computer Networks, GR-731-00 Chania, Greece

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 Quality 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 [3,20]. 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 [2]. 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 [2] 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 pre recorded 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 ($P_{\text{CR}}$), 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 $P_{\text{CR}}$ defined as follows:

$$P_{\text{CR}}^j_n = \frac{D_{\text{CR}}^j_n - T_j \cdot L_{\text{CR}}^j_n}{T_j},$$

(1)

where, the losses counter ($L_{\text{CR}}^j_n$) 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 ($D_{\text{CR}}^j_n$, 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 ($\text{Buffer}^j_n$). $\text{Buffer}^j_n$ 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 [2], 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.