Dynamic Real-Time Bandwidth Sharing Algorithm for Broadband Multimedia Communication Systems

Yacine Atif
Department of Computer Science
United Arab Emirates University
Al-Ain 17551, U.A.E
Yacine.Atif@uaeu.ac.ae

Abstract

Distributed multimedia information systems take the form of client-server architecture where the server acts as a repository for multimedia documents, which are compound documents composed of synchronized media objects. When transmitted across the network by the server, such documents require Quality of Service (QoS) guarantees to ensure a certain level of Quality of Presentation (QoP) at the client side. While broadband networks such as ATM provides end-to-end QoS guarantees, through dedicated network channels, a judicious transmission scheduling algorithm is required to decide which channel should transmit which media object and when, to achieve the required level of QoP. In this paper, an efficient real-time scheduling algorithm for delivering multimedia documents on broadband networks is discussed. This server-based scheduling algorithm dynamically adjusts itself to environmental parameters such as the available network resources and the size and the quantity of the requested multimedia documents. The technique we propose automatically controls and allocates the algorithm run-time cost, in order to minimise the transmission deadlines violation of media objects at due to the algorithm's overhead.

1. Introduction

Emerging technologies in fibre-optic and digital broadband network, such as Asynchronous Transfer Mode (ATM) and gigabit Ethernet have led to the evolution of distributed multimedia information systems (DMIS). DMIS allows applications to transmit heterogeneous traffic types such as video, audio, images and text. In such systems, a multimedia database server acts as a repository of multimedia documents, which are composite documents, composed of heterogeneous multimedia objects. These objects are temporally linked to each other according to some synchronization requirements as shown in Figure 1.

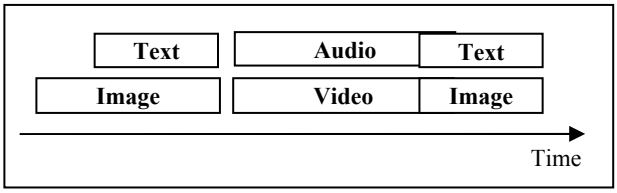

Figure 1. Multimedia document composed of synchronized multimedia data

Thus multimedia documents are pre-orchestrated documents, which are accessed by clients to be viewed locally. Note that clients do not wish to download the whole document to their local disk, as clients' computers may not have the storage capacity of the server. Clients may however buffer the current segment of the multimedia document being played.

The playback of multimedia documents at the client side should maintain the synchronization requirements within that document for a smooth display of its content. This constraint will only be satisfied if multimedia data that compose the multimedia document reach the client side at or before their scheduled playback time. Such constraints raise several problems, requiring a real-time operating system support and Asynchronous Transmission Mode (ATM) networks which distinctive QoS feature is

the provision of traffic contract through appropriate bandwidth to different types of traffic.

In this paper, we propose a dynamic scheduling algorithm which runs at the server side with the objective of assigning real-time SIUs to broadband network channels. The proposed algorithm plans the scheduled times of SIUs' transmission and the transmitting channels. Once a schedule has been built, it is delivered to the ATM subsystem layers for transmission. The delivered schedule is correct in the sense that no SIU will miss its playout deadline when it reaches the client side. The remaining sections of the paper are organized as follows. Section 2 states the problem and set the objectives of the research work discussed in this paper. Section 3 reveals our scheduling algorithm. Section 4 provides a performance evaluation of the algorithm and finally Section 5 concludes the paper with a summary of results and a proposed future work.

2. Problems and objectives

The objective of the proposed scheduling algorithm in this paper is to preserve the Quality of Presentation (QoP) requirements for quality playback of a multimedia document using the QoS guarantees of a broadband network. The proposed algorithm in this paper strives to map the user-requested QoP to the available network QoS. If the available QoS cannot accommodate the requested QoP, connection is refused. Similarly, if QoP requirements cannot be met, the document playback is interrupted. QoP is expressed in terms of threshold parameters which value need to be satisfied in order for a playback of the corresponding multimedia document to take place. For continuous media for instance, depending on the required quality, a user can quantify the acceptable data loss in a network environment. A fraction of the media objects can be dropped without degrading the required playback quality. This fraction consists in the ratio of the required rate of presentation to the nominal one. For example, if the user can tolerate a presentation rate of 20 frames per second for a video object (instead of 30 frames per second for NTSC quality video), every third frame can be dropped. A set of QoP parameters to specify the desired quality of presentation of multimedia information were defined as shown below. Table 1. Table 1 QoP parameters

This means that, in an audio clip, 1 audio SIU can be dropped for every 50 audio SIUs, and in a video clip, 1 video SIU can be dropped for every 10 video SIUs. The above QoP parameters may vary from application to application especially with respect to the content of the video SIUs. To guarantee their satisfaction, QoP parameters requirements need to be mapped to the network QoS parameters.

Each SIU_i to be transmitted on channel C_j is characterised by the size of the SIU s_i and deadline d_i. We define an SIU-to-Channel assignment (SIU_i C_j) to be feasible, if SIU_i transmitted on channel C_j reaches its destination before its playout deadline. An SIU that is feasible on one channel may not be feasible on another channel, even if this SIU is the only one assigned to the channel for transmission. This may be due to the channels' bandwidth or the size of the SIU. If a schedule includes all the SIUs, then it is considered to be complete. Otherwise, it is referred to as a partial schedule.

3. Dynamic real-time scheduling

Scheduling can be represented as the problem of incrementally searching for a feasible schedule in the graph $G(V,E)$, that represents the solution space [2]. G in our representation of the problem is in the form of a tree as shown in Figure 2. The vertices $v_i \in V$ in G represent SIU-to-channel assignments (SIU_i,C_j). A partial path (v_i, v_j, …,v_k) from the root v_i to a vertex v_k in G represents a partial schedule. The partial schedule includes all the SIUs that have been scheduled so far and is represented by v_i, v_j, …,v_k. The edges represent extending the partial schedule by one more SIU-to-channel assignment. To extend a schedule (or path), one SIU is assigned to a channel at a time, and the feasible schedule is incrementally built. Thus, schedule construction proceeds from one level in G to the next level. Complete schedules, if they exist, will be at the leaves of G. The incremental schedule construction allows a dynamic algorithm to produce a partial schedule that is feasible at any point during scheduling process. To choose a new SIU to add to the schedule and to assign that SIU to one of the channels, we need to evaluate and compare a set of candidate vertices with one another in G. The candidate vertices are stored in a candidate list, CL of all feasible SIU-to-Channel assignments for the SIUs considered so far. Thus, feasibility checks are applied to different vertices in G to identify the valid SIU assignments that can be added to the partial schedule. The search makes use of a heuristic function, which prioritizes the nodes to be explored based on SIUs' Earliest Deadlines at the SIU dimension and channels' Earliest Available Time at the channel dimension.

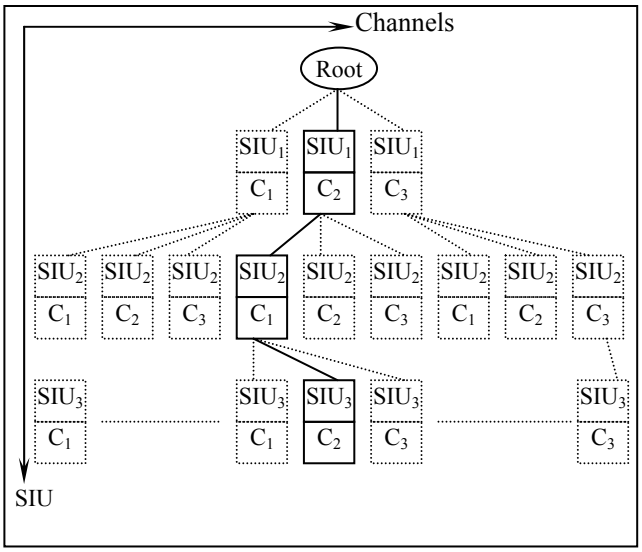

Figure 2 Search-based representation of scheduling

In this algorithm, SIUs arrive continuously and form the input of the scheduler, which selects the best assignments SIU-to-channel on-line, and distributes the SIUs to their corresponding channels in the system for transmission. Hence, the scheduler switches between two states: a scheduling phase followed by SIUs' transmission phase. The scheduler runs in parallel with the transmission of SIUs by the network channels. Thus, while the network channels are transmitting the batch of SIUs from the previous phase, newly arrived SIUs are scheduled and later they are added to the transmission queues of the assigned channels as shown in Figure 2.

The time at which the problem results are produced is critical in any real-time systems. The time consumed to produce those results is equally important especially in dynamic environments [2]. The more time spent in building a schedule, the better will be the quality of the resulting schedule. However, the delay in delivering the schedule may result in missing the deadlines of scheduled SIUs. To resolve this dilemma, we propose in this section a time-limit generation formula based on which the scheduling overhead is upper-bounded to avoid missing SIUs' deadlines because of scheduling time. As such, we have factored in a scheduling quantum $Q_s(j)$ (i.e. time-limit) in the scheduling algorithm that changes dynamically in reaction to various parameters such as slack time, SIUs arrival rate and actual channels' load, which affect the value of the time to allocate to a scheduling phase. For instance, a large slack-time, a low arrival-rate and a high channels' load suggest a large quantum value. When the channels load is high, even if the scheduler is able to produce a feasible schedule quickly, the channels are not able to transmit the assigned SIUs immediately. On the other hand, a short slack-time, a high arrival-rate and a low channel load suggests a small value for the time quantum. By bounding the scheduling time, the algorithm ensures that SIUs do not miss their deadlines because of scheduling overhead. The criterion to control the allocation of time quantum $Q_s(j)$ of a scheduling phase j is shown in Figure 3.

$$Q_s(j) <= \min(\max(min\_slack, min\_load), k/\lambda)$$

where: $min\_slack = \min(Slack_i \mid SIU_i$ element of $batch(j))$

$min\_load = \min(Load_k \mid C_k$ element of channels))

Figure 3.. Criterion for allocation of scheduling-time.

The expression in Figure 3 considers the maximum of *min_slack* and *min_load*. By taking *min_slack* into consideration, we ensure that no SIU will miss its deadline due to scheduling overhead. However, if the set of channels are not able to transmit the SIUs upon successful delivery of the schedule due to their current load, then these SIUs' deadlines are going to be missed anyway even if they are assigned immediately. In this situation, the value of the time quantum is extended to maximize the quality of the schedule.

4. Performance evaluation

A comparative study in the context of a simulated set of experiments evaluates the performance of the proposed algorithm. The candidate algorithms are the time-constrained search algorithm discussed in this paper and the forward algorithm proposed in [1]. Two versions of the search algorithm were implemented. The first version labelled SA in the performance results is a pure search algorithm, which treats all dead-ends as hard dead-ends. The second version of the algorithm labelled SA-SD employs the selective dropping technique which differentiates between between various types of traffic to comply with the user's QoP requirements. The motivation behind the implementation of the two versions is to evaluate the effect of the selective dropping technique employed by the search algorithm. The other candidate algorithm, namely the forward algorithm and labelled in the figure FA works as follows. SIUs are first ordered in an increasing order of their playout deadlines. Each SIU is then scheduled on the channel that results in the earliest transmission time. The forward algorithm belongs to the best effort class of algorithms by opposition to our

algorithm, which is feasibility-based, and feasibility checks are performed at the server side. The experiment results were obtained following 10 runs of the simulated algorithms. The mean of the 10 runs was plotted in Figure 4.

The problem set consists in a set S of Poisson-based distributed SIUs. The performance metric is the percentage of SIUs reaching the client side before their playout deadlines. Several experiments were conducted but due to space limitation, we show below the result of a single experiment showing the rate of successfully transmitted SIUs at the y-axis versus the deadline factor at the x-axis which low values reflect tight SIUs' deadlines whereas high values correspond to loose SIUs' deadlines. SIUs' sizes are uniformly distributed between 0.1 Kbytes and 6 KBytes. The deadline, d_i, of SIU_i is uniformly distributed within the interval (End_i, D_{max}) measured in milliseconds, where End_i is the worst-case transmission time of SIU_i (i.e. assuming it uses the channel with the lowest bandwidth). D_{max} is calculated as follows: D_{max}= Deadline_Factor × End_i where Deadline_Factor is a parameter in our experiment that controls the degree of laxity in SIUs' deadlines. Larger Deadline_Factor represents larger slack times, whereas small Deadline_Factor represents tight deadlines. The network resources consist in four channels with bandwidth 1, 2, 3 and 4 Mbps respectively. Finally, note that in our experiment, the scheduling cost for the Forward algorithm is set to 0, whereas the search algorithm is penalized by its incurred time-complexity which is however maintained under control as discussed earlier in this paper.

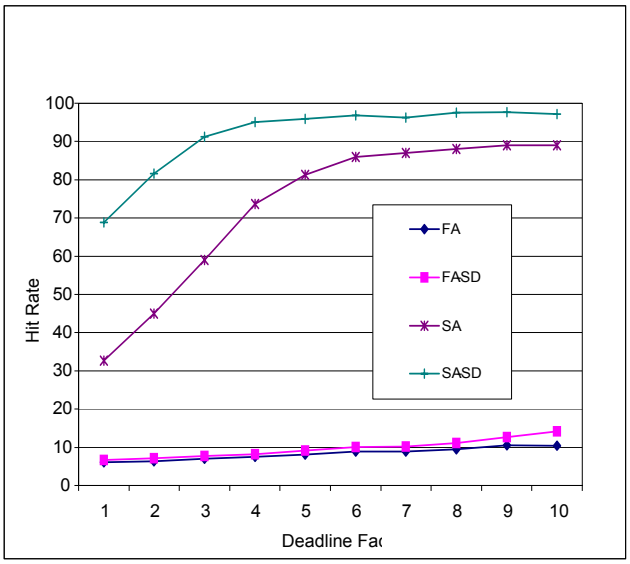

Figure 4. Transmission rate(%) vs. Deadline Factor on 4 network channels with bandwidth 1, 2, 3 and 4 respectively.

As shown in Figure 4, the proposed search-based algorithm outperforms the forward algorithm in terms of SIUs' deadline hit-rate by as much as 80% at loose deadlines. This means that the search-based algorithm takes better advantage of deadlines looseness in maximizing the transmission rate of SIUs. Such performance gain by the search algorithm is attributed to the fact that the search algorithm checks for deadline feasibility in the schedule whereas the Forward algorithm is only based on channel availability times. Also, the forward algorithm does not scale-up performance as much as the search based algorithm does when equipped the selective dropping mechanism. This is attributed to the fact that SA-SD algorithm progresses faster towards a feasible schedule than SA algorithm since in SA-SD, backtracking is attempted only when hard dead-end are encountered during the search process, whereas SA backtracks to the previous search level whenever an SIU cannot be scheduled on any of the available channels. Such backtracking, delays the scheduler from moving deeper in the search space towards a leaf node, which reflects a complete feasible schedule. The margin of performance of SA-SD over SA is about 40%. This is indicative that the selective dropping technique which makes use of the users' QoP requirements is employed efficiently by the search algorithm especially at tight deadlines, where backtracking situations are often encountered.

5. Conclusion

In this paper, a dynamic non-pre-emptive real-time scheduling algorithm for high-level multimedia synchronisation on broadband networks has been proposed. The algorithm runs at the server side above the ATM layers to negotiate the transmission of media objects composing multimedia documents so that they reach the client side before their playout deadlines.

6. References

[1] Shahab Baqai, M. Farrukh Khan, Miae Woo, Seiichi Shinkai, Ashfaq Khokhar, and Arif Ghafoor, *"Quality-Based Evaluation of Multimedia Synchronisation Protocols for Distributed Multimedia Information Systems"*, IEEE Journal on Selected Areas in Communications, Vol. 14, No. 7, pp1388-1403, 1996.

[2] Atif Y, "*Dynamic Load-Assignment in Distributed Memory Multiprocessors*", International Journal of High-Speed Computing, vol. 10, No. 1, pp. 83-113, 1999.