Distributed Scheduling of Recording Tasks with Interconnected Servers*

Sergios Soursos[1], George D. Stamoulis[1], and Theodoros Bozios[2]

[1] Department of Informatics, Athens University of Economics and Business
{sns, gstamoul}@aueb.gr
[2] New Technologies Department, INTRACOM S.A.
tmpo@intracom.gr

Abstract. We consider a system with multiple interconnected video servers storing TV programs that are received through satellite antennas. Users, equipped with set-top boxes, submit requests for TV programs, to each of which they assign a utility value according to their preferences. We develop a distributed scheduling algorithm that selects the programs to be recorded and the servers to store them, so that a high total utility is generated to the users' population. Our scheduling algorithm is based on the programs' broadcasting information, the users' preferences, the constraints regarding the capabilities of simultaneous recordings and storage, and the system's topology. In fact, servers belonging to the same cluster co-operate in order to attain increased efficiency by exchanging content through streaming or replication. The efficient performance of our scheduling algorithm is shown by means of experiments. The algorithm constitutes a practically applicable solution, already implemented and integrated in the testbed of the IST project UP-TV.

1 Introduction

The technology of digital television offers new possibilities for personalization and optimization of services that cannot be provided by analog broadcasting technologies. Additional information (i.e. metadata) concerning classification of content, starting and ending times etc. can be included in the digital broadcast stream and can greatly help in scheduling the recording and broadcast of the various programs so as to optimize certain performance indices. These issues were investigated by IST project UP-TV (**U**biquitous **P**ersonalized Interactive Multimedia **TV** System and Services, IST-1999-20751) [6]. The purpose of UP-TV was to create advanced and expandable architectures and systems for TV Anytime applications, thus providing personalized access to broadband information in an interactive way.

* This work is partly funded by the project IST-1999-20751 UP-TV whose partners are: INTRACOM S.A.(prime contractor), Universitat Paderborn, Invonova Gmdh, Pixelpark, PPS (Press Programm Service), Grundig, ERT (Hellenic Broadcasting Corporation), TUC (Technical Univ. of Crete), NV TV Limburg, and AUEB (Athens Univ. of Economics and Business) as a subcontractor of INTRACOM S.A.

N. Mitrou et al. (Eds.): NETWORKING 2004, LNCS 3042, pp. 1483–1488, 2004.

In this paper, we present the algorithm employed by the distributed Scheduler module for the purpose of scheduling the recording, the replication and the streaming of TV programs within the UP-TV network of servers, taking into account the preferences of the users and the various limitations regarding the capabilities of simultaneous recordings and storage, and the system's topology. The objective of the module is to generate a schedule leading to a high total utility for the users' population. This problem is very complicated to solve optimally for a realistic system with multiple servers, many different channels and numerous users. Therefore, we resort to a heuristic approach and develop a practically applicable algorithm. In fact, our approach does not only apply to the "simple" one-shot case, where we have to decide on the recording schedule once and for all. On the contrary, information on future TV programs can be received (as input) by the Scheduler *at any time instant.*

The problem addressed in this paper is similar to that of filling caches and scheduling content placement in Content Distribution Networks. For example, certain replication heuristics are presented and evaluated in [1], while several replica placement algorithms are developed and evaluated in [2]. However, contrary to our paper, such works are not dealing with content placement in conjunction with scheduling of program recordings.

2 The UP-TV Environment

The UP-TV distributed environment consists of several *clusters* which, in general, are interconnected through low bandwidth connections (e.g. through the Internet). A cluster is formed by a high-bandwidth network of UP-TV enabled servers (see Fig. 1). A number of users within a certain geographic area or a certain organization are served by the cluster's servers, through ADSL or CATV connections. Every server in the cluster has the following components, each of which poses restrictions to the Scheduler's algorithm: i) one or more hard disks for storing TV programs, with total capacity C, ii) a number R of satellite receiver cards that can record simultaneously different programs from different channels, iii) a number A of satellite antennas, each receiving the broadcast signal from one satellite, and iv) a number L of Low-Noise Block downconverters (LNBs) that can lower a signal's frequency and at the same time isolate the desired quarter of the signal's band, which then feeds a satellite receiver card. Note that it only makes sense to select L so that $L \leq 4A$ and $L \leq R$.

3 The Distributed Scheduling Algorithm

Within the cluster, all the servers run a local Scheduler module and one of them runs both the local Scheduler and the cluster's Central Scheduler (CS). The distributed scheduling algorithm consists of three parts. In Part 1, each local Scheduler decides which program will be selected for recording based on the preferences of the local users and the local technological limitations. In Part 2, each server sends its decisions to the CS; then, to each program which is to be

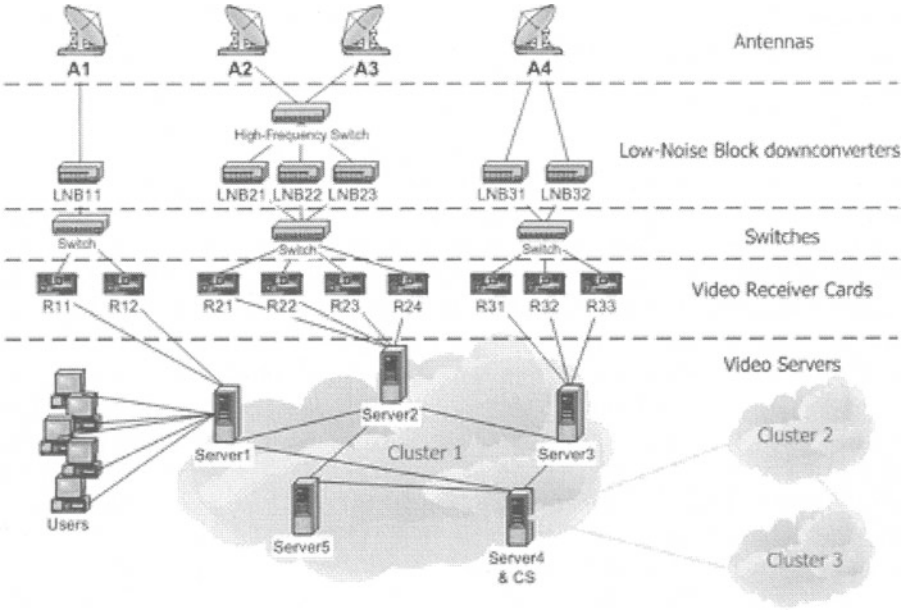

Fig. 1. The UP-TV environment

recorded by multiple servers the CS assigns a server which should certainly record it. Finally, in Part 3, the decisions of the CS are sent back to the local Schedulers, along with the information about the cluster's topology; each Scheduler then checks if it is beneficial to store locally programs that are assigned to other servers (in Part 2) or retrieve them when needed by means of streaming.

In our approach, each program is assigned a utility value that represents the total satisfaction to be generated to all users to which the program is of interest. Note that the system gives each user the opportunity to select his preferred programs and assign to them a utility value. (For each program, the utilities assigned thereto by the various users are summed.) The objective of the local Scheduler (i.e. Part 1 of the algorithm) is to compute recording schedules that will lead to a high *total satisfaction* (i.e. total utility) of the users, subject to the aforementioned limitations on the basis of the program's broadcasting information and the users' preferences. Such a maximization problem is very hard to solve exactly. (Even simplified versions of the problem reduce to the knapsack problem, which is NP-hard [3].) Therefore, we resort to a heuristic approach, the main ideas of which are outlined below. The key idea of our algorithm is that the profitability of a program depends on both the utility that it will generate and on the size of disk space it will occupy. Therefore, a proxy of the profitability of a program expressing the aforementioned trade-off, is the *Utility per Mbyte*(UpM); i.e., the ratio of the utility and the required disk space. The algorithm then considers for recording the programs *one-by-one* in decreasing order of UpM, while conforming to the constraints on availability of

disk space, satellite receivers and LNBs. This approach can be motivated by considering each program as an individual bidder in a *sealed-bid auction*, who is bidding an amount of money equal to its own utility; e.g. see [4]. Another critical issue is the efficient exploitation of LNBs. A proxy representing the trade-off between the value introduced by a program's recording and the "cost" due to the time of occupying an LNB is the Utility per Second (UpS). However, this is almost equivalent to UpM, since the coding rate of the TV channels' program will be the same (or almost so) for different channels.

An important feature of our algorithm is that it does not only apply to the "simple" one-shot case, where we have to decide on the recording schedule once and for all. On the contrary, information on future TV programs can be received (as input) by the Scheduler *at any time instant* and previous information (i.e. programs' utilities) is updated. Future recording decisions are ignored, while recording decisions in progress are not subject to change, for simplicity reasons. Already recorded programs may have to be deleted, in order to save disk space for recording more profitable programs. The utility of each such program diminishes exponentially as time elapses. If a new program is decided to be recorded but there is not enough disk space available, then the Scheduler checks if it can gain the required capacity by deleting one or more recorded programs, if beneficial.

In Part 2 of the distributed algorithm, the clsuter's Central Scheduler (CS) receives the local decisions of all servers. Except from their local decisions (i.e. a sorted program list in decreasing order of UpM), all the servers of the cluster send also a few high-value programs that could not be selected for recording, due to satellite receiver or LNB unavailability. In cases where more than one cluster's servers have decided to record the *same* program, the CS decides which one of the servers should certainly record the program, based on the ranking that each server gave to the particular program. Thus, the CS gives the other servers the option of either i) storing locally the program or ii) streaming it from the server decided by the CS. Finally, the CS sends its own decisions back to the local Schedulers of the other servers together with the cluster's topology, which includes all the links of the cluster along with the link's capacity. Also, the CS sends the location of the high-value programs that couldn't be recorded, giving the option of duplication to the interested servers. In Part 3, each local Scheduler must decide how to handle the programs that were initially selected and the CS has assigned their storage to other servers. Motivated by [5], we use two metrics for evaluating the cost for storing a TV program to the disk and the cost for streaming it from another server via the network; namely, the *storage cost per unit* and the *streaming cost per unit*, which can be assigned proper values on the basis of current prices of hard disks and leased lines. The total storage and streaming costs for a specific TV program can be calculated as follows:

- *total storage cost* =
  (storage cost per unit) $*$ *(program storage time)* $*$ *(program storage size).*
- *total streaming cost* =
  (streaming cost per unit) $*$ *hops* $*$ *(program storage size)* $*$ *requests* $*$ 0.6,
  where the factor 0.6 accounts for the fact that due to caching, not every new

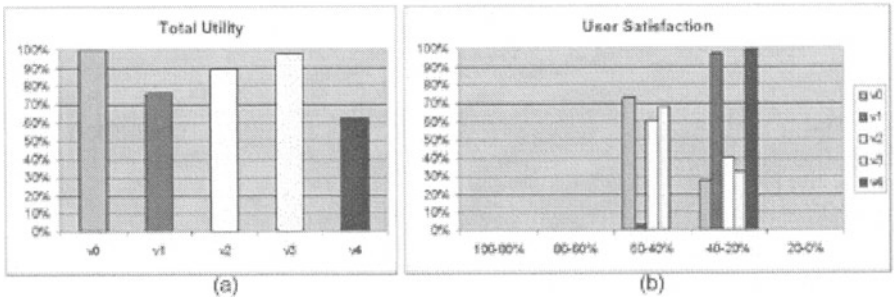

Fig. 2. Comparison of variations. v0 stands for the original algorithm, v1 for its first variation etc.

local request for a specific program causes a new streaming session thereof, and thus it does not add to its total streaming cost.

Each local Scheduler finds the shortest path and computes the *total streaming cost* and the *total storage cost* for each program that was initially decided to be stored locally and the CS has assigned its storage to another server. Comparing the above costs will lead the local Scheduler to a decision of streaming, if the former cost is lower than the later and at the same time the amount of bandwidth necessary for streaming is available in all links of the path.

After making all of the above decisions for leaving or not programs for streaming, each local Scheduler checks if any disk space has been released due to cancellation of recording decision(s). Then, it checks one-by-one (in decreasing order of UpM) the high-value programs that couldn't be recorded due to unavailability of satellite receivers or LNBs, and decides which ones to replicate from other servers. Finally, it re-executes Part 1 of the distributed algorithm, in order to take advantage of any unused disk space by recording additional programs.

4 Experimental Results

As already explained, the problem dealt is extremely complicated to solve exactly. Thus, in order to assess how efficient the proposed heuristic is, we must provide considerable evidence that other less complicated heuristic methods provide inferior results. Therefore, for the evaluation of the Scheduler's algorithm, we have defined the following simpler variations of the algorithm:

- Under the first two variations, the programs are sorted in Part 1 according to a different metric than UpM: i) sorted in descending order of the total utility and ii) sorted in ascending order of the required disk capacity.
- The third variation does not comprise Parts 2 and 3 of the original algorithm.
- The fourth variation is a totally greedy algorithm where the programs are sorted in ascending order of their starting times, while Parts 2 and 3 of the original algorithm and other optimization heuristics are not included either.

We have defined certain performance metrics that are appropriate for the comparison of the original algorithm with the above variations:

- *Total Utility:* the sum of the utilities of the recorded programs for all the servers in the cluster; streamed and duplicated programs are also included for the variations comprising Parts 2 and 3.
- *User Preferences:* the percentage of users of the cluster that have access (either locally or via streaming) to their first choice of a program, to their first *and* second choices, and to their first *or* second choices.
- *User Satisfaction:* the percentage of users of the cluster belonging to each specific level-of-satisfaction group (namely, 100%-80% group, 80%-60% group, etc.), where for each user we compute the fraction of the utility of all programs to which the user has access divided by the utility of all programs requested by him.

A realistic broadcast schedule was employed in our experiments. Users' preferences were generated randomly. A star topology was taken for the UP-TV cluster, with a router at the center and three servers at the edges. The bandwidth of each link was taken 100 Mbps. We have conducted many experiments, with different setup. In Fig. 2, we present some indicative results. In particular, our algorithm out-performs the rest variations in terms of total utility, with the total-greedy algorithm having the lowest one [see Fig. 2(a)]. The use of simpler metrics for sorting definitely leads to performance degradation. Considering the users' preferences (not depicted here), only the first variation is close in performance with the original algorithm. But, when it comes to the users' satisfaction [see Fig. 2(b)], it is clear that our algorithm has the best performance since it satisfies to a relatively high degree a high percentage of users.

In general, experiments have revealed that the distributed scheduling algorithm outperforms its variations with respect to the various performance criteria. Given also its relatively low computational complexity, it appears that this algorithm constitutes a promising and practically applicable solution.

References

1. J. Kangasharju, J. Roberts, K.W. Ross, "Object Replication Strategies in Content Distribution Networks," Proceedings of WCW '01: Web Caching and Content Distribution Workshop, Boston, MA, June 2001.
2. L. Qiu, V.N. Padmanabhan, G.M. Voelker, "On the Placement of Web Server Replicas," Proceedings of the 20th IEEE INFOCOM, 2001.
3. M.R. Garey, D.S. Johnson, "Computers and Intractability - A Guide to the Theory of NP-Completeness," W.H. Freeman and Company, ISBN: 0-7167-1045-5.
4. L.M. Ausubel, P. Cramton, "Demand Reduction and Inefficiency in Multi-Unit Auctions," working paper, University of Maryland, 2002.
5. S.-H. G. Chan, F. Tobagi, "Caching Schemes for Distributed Video Services," Proceedings of the 1999 IEEE ICC, Vancouver, Canada, June 1999.
6. Project IST-1999-20751 UP-TV, URL: `http://www.up-tv.com`