Optimizing Continuous Media Delivery by Multiple Distributed Servers to Multiple Clients using a Genetic Algorithm

Gerassimos Barlas and Khaled El-Fakih

Department of Computer Science,
American University of Sharjah, P.O.B. 26666, UAE
{ gbarlas, kelfakih }@aus.ac.ae

Abstract. In this paper we explore the potential of a VoD system that is based on a paradigm that has been recently proposed: that of combining many distributed servers to handle the delivery of each requested media. When faced with a system comprised of N servers and M clients, the problems to be addressed are: (i) how to split the delivery task among different servers and (ii) how to pair clients and servers, the objective being to minimize the access time of the clients.
To this end, we present an analytical framework that enables the division of the delivery process for each client in a distributed manner. This framework is coupled with a genetic algorithm that enables an optimal or near-optimal solution to the problem of pairing clients and servers, in a small number of generations.
The paper is concluded by a rigorous study of a N-servers, M-clients system that answers a number of important questions like, what is the quality of service achieved and how our proposed system behaves under increased load.

Keywords: multiple servers, multiple clients, genetic algorithms, video on demand, divisible load.

1 Introduction

VoD presents a number of significant problems that have limited the scope and clientele of most deployed systems to-date. The size of the data involved, even with the latest codecs available (MPEG4), presents serious challenges to the video servers and the carrier network. Multicasting [6] and other technologies like simulcasting [7] permit the shift of the bottleneck from the servers themselves. Alas, these delivery approaches also "shift" control away from the end-user which is ultimately the main allure of the VoD paradigm.

Keeping up with the user demands, puts hefty requirements on the servers. Parallel video servers and intelligent disk scheduling has the potential needed, but more-often-than-not they offer a localized solution. Clients with a slower connection to a server have to wait for a very long time before the playback can commence (a.k.a. Access Time - AT).

A recently proposed approach to the delivery of continuous documents is to employ multiple *distributed* servers to deliver a requested document. The framework needed for scheduling such a delivery has been originally presented in [20] and further expanded and refined in subsequent publications [3] for the case of a single client. In this paper we address the problem of how a system with multiple clients and servers behaves and what are the characteristics of the services offered. Our contribution also lies in the proposal of a genetic algorithm for solving the server-client mapping problem under the constraint of minimizing the average access time.

As a first step we describe the mathematical framework that enables the calculation of a multi-server delivery schedule for a single client. Both single and multi-installment strategies are described, the latter offering better utilization of available resources, smaller AT and better suitability to adaptive strategies. The genetic algorithm works by randomly producing a population whose individuals are possible solutions to the client-server mapping problem. The algorithm produces optimal (or near optimal) mappings without suffering from intractability. Our genetic algorithm is hybridized, i.e. departs from classical GA techniques, by employing procedures for finding feasible schedules and for avoiding premature convergence to local optima. We have used a GA since we do not have an exact solution for the delivery strategies and since GAs were successfully used in solving a variety of optimization problems [8][15].

Our simulation study shows that the proposed solution offers a vastly improved level of service from the traditional single server/per client approach. Not only are average AT substantially lower, but even under extreme loads that would cause denial of service to some clients, our system deteriorates gracefully. Please note that the term server is used here to refer to server sites and not particular machines. Each site could be a parallel server farm.

The paper is organized as follows: Section 2 presents related work in the field. Section 3 holds a formal definition to the problem followed in Section 4 by the mathematical framework needed for computing how to service a single-client's request. The genetic algorithm used to compute a near-optimum schedule for a battery of clients is presented in Section 5. A simulation study concludes the paper by comparing our work to the classical single-server per client approach.

2 Related Work

The idea of combining multiple geographically distributed servers for video data delivery was originally introduced in [20]. The idea of using multiple connections to geographically distributed mirror sites has been also suggested by Rodriguez et.al. for minimizing FTP download times [18]. In similar premises, the GridFTP protocol extension and associated tools facilitate speedier object replication/mirroring in high-performance computational grids [19]. GridFTP supports both parallel and stripped transfers , the latter being closer to the technique utilized by our multiserver distribution strategy. In this paper we assume only single connections for each client-server pair, although multiple connections can be supported without modifications to the partitioning framework of section 4.

Most of the work on VoD has been focused on parallel video servers. A good introduction on the subject is given by J. Lee in [14]. Jung et.al [12] have also proposed the use of a data partitioning scheme among servers that resembles data-interleaving in disk arrays (RAID).

Multicasting [6] can maximize network utilization while at the same time minimize the server load. Patching has been proposed by several researcher as a way of accommodating clients with different playback-starting times - a major obstacle in the use of multicasting for VoD. Patching minimizes communication overheads by retransmitting only the parts of the movie that a client has missed [5]. The problem with multicasting is that it requires infrastructure changes like special routers, etc.. Also, traffic localization is needed in order to cope with network errors and communication delays. Not to mention of course, that each movie should be requested by many clients at the same or close-enough time.

The simulcast protocol [7] has been proposed as an alternative to multicasting. The advantage is that no modification or special treatment by the network is required. Simulcasting uses the clients as repeater nodes. The problem is that asymmetrical connections like ADSL are not designed to efficiently support such operations, making simulcast a choice only for low bit-rate content.

Rejaie et.al. [17] proposed a layered video format and associated mechanism for tuning data volume to network capacity. Pejhan et.al. [16] have demonstrated a similar technique adapted to the MPEG standard. By storing a single video along with the motion vectors needed for encoding it at smaller frame rates can be efficiently used for quality adaptation. Both techniques allow the adaptation of the QOS to network congestion and/or different communication speeds. However, compromising quality can be a downside to a commercial service.

3 Problem description

The architecture examined in this paper consists of N servers connected to M clients by generally non-uniform connections via the Internet. Each server holds complete copies of the movies that could be requested.After a client requests a particular movie, a number of servers $\leq N$ are devoted to serving this request, by sending *disjoint* parts of the requested movie to the client.

Each server S_i devotes $bw_{i,j}$ bandwidth for servicing client C_j. The number of connections that can be simultaneously activated to a client are limited by the client's available bandwidth bw_{C_j}. Similarly, a server cannot accept connections that would exceed its total bandwidth bw_{S_i} (see section 4.3). In the remaining sections bandwidth is expressed in time-per-data-transferred units (e.g. sec/MB)

The arrangement described above for the distribution of the media files is a proxy-at-client architecture [14], i.e. each client has to merge the individual server streams prior to playback. An alternative approach could be to have the proxy implemented by an ISP, thus making the client totally agnostic to the details of the implementation and thus far simpler.

In order to build a model of the whole process, we assume that an affine model dictates the communication costs. Thus, sending a part m of a movie from server S_i to client C_j requires $bw_{i,j}\ m\ L_j + o$, where o is a constant overhead that can be associated with setup activities or the cost of establishing a connection etc., and L_j is the length of the movie requested by client C_j. In this paper we assume that o is incurred only once for each request and that it is the same for all servers.

The problem of a single client with multiple servers has been examined before in the literature [20], where a thorough comparison of single and multi-installment strategies is available. The derivations presented in subsections 4.1 and 4.2 are simplifications of work presented in [3]. In this paper we focus on examining the behavior of a large scale system involving many clients and servers using a multi-server delivery strategy. Such a system can offer improved customer service while maximizing server utilization. Our model simplifications may introduce a level of inaccuracy as far as real systems are concerned (for example data losses are ignored) but these issues can be addressed in the future.

In the remaining sections we also use these notations: R represents the inverse of the movie playback rate (expressed in sec/MB). R is assumed to be constant for each movie, i.e. we have Constant Bit-Rate media. $m_{j,i}$ represents the document part that is delivered by server S_i during installment j. For the next subsection only, m_i is used instead of $m_{0,i}$ for simplicity.

4 Scheduling Document Delivery

4.1 The Single Installment Case

If we assume that the clients don't share any communication links apart from the ones originating from the servers, then the delivery schedule can be found for each individual client separately.

To simplify notation, in this subsection we remove the index related to client identification from our parameters. If $k \leq N$ servers upload content to a client, continuity of the playback is ensured if the following hold:

$$AT + m_0\ R\ L \geq m_0\ L\ bw_0 + o \tag{1}$$

$$AT + (m_0 + m_1)\ R\ L \geq\ m_1\ L\ bw_1 + o \tag{2}$$

$$\cdots$$

$$AT + R\ L\ \sum_{i=0}^{k-1} m_i \geq\ m_{k-1}\ L\ bw_{k-1} + o \tag{3}$$

In order to minimize the AT, the equality sign should be used in the above inequalities. By doing so, and by subtracting successive inequalities, we get $m_{i+1} = \frac{bw_i\ m_i}{bw_{i+1} - R}$ for $i = 0, \ldots, k-2$. Or in general

$$m_{i+1} = \frac{m_0 \prod_{l=0}^{i} bw_l}{\prod_{l=1}^{i+1} (bw_l - R)} \tag{4}$$

Since the sum of all m_is should be equal to 1 (also referred to as the normalization equation), then we can compute m_0 from:

$$\sum_{i=0}^{k-1} m_i = 1 \Rightarrow m_0 = \left(1 + \sum_{i=1}^{k-1} \frac{\prod_{l=0}^{i-1} bw_l}{\prod_{l=1}^{i}(bw_l - R)}\right)^{-1} \tag{5}$$

and the access time from:

$$AT = L\,(bw_0 - R)\,m_0 + o \tag{6}$$

In the case of uniform connections $bw_i \equiv bw$, the above equation can be used for estimating the number of servers that would be needed given a desired AT:

$$(5),(6) \Rightarrow \left(\frac{bw}{bw - R}\right)^k = \frac{R\ L}{AT - o} + 1 \Rightarrow k = \lceil \frac{log(\frac{R\ L}{AT-o} + 1)}{log(\frac{bw}{bw-R})} \rceil \tag{7}$$

4.2 The Multi-Installment Case

Server utilization can be substantially improved by splitting the document delivery into several installments, as each server stays idle for less time. Using a multi-installment approach means that each server uploads disjoint document parts in sequence (see Figure 1).

The continuity of the playback given N servers and W installments, is ensured by satisfying the following $N \cdot W$ inequalities for each j installment, i server pair:

$$AT + R\ L\ \left(\sum_{l=0}^{i} m_{j,l} + \sum_{k=0}^{j-1}\sum_{l=0}^{N-1} m_{k,l}\right) \geq L\ bw_i \sum_{k=0}^{j} m_{k,i} + o \tag{8}$$

In order to minimize the AT, the equality sign should be used in the above inequalities. By subtracting successive equalities for the parts delivered during the first installment it can be shown similarly to the single installment case, that

$$m_{0,i+1} = \frac{m_{0,0}\prod_{l=0}^{i} bw_l}{\prod_{l=1}^{i+1}(bw_l - R)} \tag{9}$$

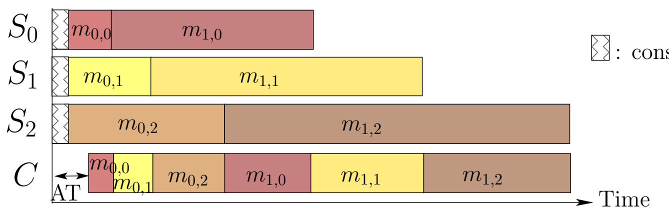

Fig. 1. Example of a multi-installment delivery by 3 servers employing 2 installments.

Using the same approach for installments $j = 1, \ldots, W-1$, we can show that:

$$m_{j,i} = \begin{cases} \frac{bw_{N-1}\sum_{k=0}^{j-1} m_{k,N-1} - bw_0 \sum_{k=0}^{j-1} m_{k,0}}{bw_0 - R}, & i = 0 \\ \frac{bw_{i-1}\sum_{k=0}^{j} m_{k,i-1} - bw_i \sum_{k=0}^{j-1} m_{k,i}}{bw_i - R}, & i = 1, \ldots, N-1 \end{cases} \tag{10}$$

Equations (9) and (10) define each $m_{j,i}$ as a linear function of parts that precede it in the play-back order. Since (9) to (10) represent linear relationships between each $m_{j,i}$ and $m_{0,0}$, computing the latter can be done by assuming that $m_{0,0}$ is a constant, i.e. 1. This is equivalent to multiplying all parts by a constant $C \geq 1$ which can be estimated by the normalization equation:

$$\sum_{k=0}^{W-1} \sum_{l=0}^{N-1} m_{k,l} = 1 \tag{11}$$

since the sum of the computed parts will equal C. The above constitutes a $O(N\,W)$ process that can yield very quickly the desired movie partitioning/schedule.

4.3 The General Case

In the general case, we have M distinct clients C_j, each requesting a different movie of L_j length and with a playback rate of R_j. The additional problem that needs to be addressed is that of determining which servers should be allocated to each client in order to minimize the average AT. To solve this mapping problem we employ a genetic algorithm that is shown to converge to a solution in a small number of generations.

All requests are assumed to be simultaneous and have to be serviced concurrently. Despite this being an unrealistic assumption, it is not without merit as it allows us to investigate the steady-state of the system. Also, the genetic algorithm we present, can be employed in an adaptive system, where the network conditions constantly change and the pairing of clients-servers has to change in response. An incremental GA [1] might be appropriate for solving this problem. In this case, the genetic algorithm could be employed for revising the distribution plan at regular intervals. The new schedule would take effect for each client at the beginning of each new installment, hence the benefit of using multiple installments. Such a modification could be easily incorporated in our algorithm.

The objective function that has to be minimized is the following:

$$Z = \overline{AT} = \frac{AT_1 + AT_2 + \ldots + AT_M}{M} \tag{12}$$

All the above are subject to the following constraints:

– Each server allocates only one slot to each client, i.e.:

$$k_j \leq N \quad \forall\, j = 0, \ldots, M-1 \tag{13}$$

where k_j are the number of servers/connections utilized by C_j.

– The total bandwidth consumed should not exceed the servers' capacity:

$$\sum_{j=0}^{M-1} bw_{i,j}^{-1} \leq bw_{S_i}^{-1} \tag{14}$$

$\forall\ i = 0, \ldots, N-1$. If a server S_i is not employed by a client C_j, $bw_{i,j} = \infty$.

– Each client's bandwidth should not be exceeded:

$$\sum_{i=0}^{N-1} bw_{i,j}^{-1} \leq bw_{C_j}^{-1}\ \forall\ j = 0, \ldots, M-1 \tag{15}$$

5 A Genetic Algorithm for the Mapping Problem

Genetic algorithms have been adopted for solving a variety of engineering, science, and operations research problems [8][15]. In the following subsection, we give background information on the genetic paradigm and subsequently we describe how a GA can be adapted for solving the mapping problem.

5.1 Background

Genetic algorithms are based on the mechanics of natural evolution [9]. Throughout their artificial evolution, successive generations each consisting of a population of possible solutions, called individuals (or chromosomes, or vectors of genes), search for beneficial adaptations to solve the given problem. This search is carried out by applying the Darwinian principles of "reproduction and survival of the fittest" and the genetic operators of crossover and mutation which derive the new offspring population from the current population.

Reproduction involves selecting, in proportion to its fitness level, an individual from the current population and allowing it to survive by copying it to the new population of individuals. The individual's fitness level is usually based on the cost function given by the problem under consideration. Then, crossover and mutation are carried on two randomly chosen individuals of the current population creating two new offspring individuals. Crossover involves swapping two randomly located sub-chromosomes (within the same boundaries) of the two mating chromosomes. Mutation is applied to randomly selected genes, where the values associated with such a gene is randomly changed to another value within an allowed range. The offspring population replaces the parent population, and the process is repeated for many generations. Typically, the best individual that appeared in any generation of the run (i.e. best-so-far individual) is designated as the result produced by the genetic algorithm.

In the following subsections, we describe how genetic algorithms can be adapted for solving the mapping problem. We present the components of a hybrid genetic algorithm (GA) for minimizing the function Z defined in Eq. (12). The algorithm is hybridized by procedures and design choices that account for both the likelihood of producing infeasible individuals as a result of crossover and mutation, and for the premature convergence to a local optima. An outline of the algorithm is given in Fig. 2.

```
Generation of initial population, Size POP
Set rates for genetic operators
Evaluate fitness of individuals
Repeat
    Rank individuals & allocate reproduction trials
    for(i=1 to POP step 2) do
         Randomly select 2 parents from list of reproduction trials
         Apply crossover & mutation
    Endfor
    Evaluate fitness of offspring's
    Check feasibility of individuals
    Do hill-climbing
    Preserve the fittest-so-far (elitism)
Until (termination criterion is satisfied)
Solution = Fittest.
```

Fig. 2. Hybrid Genetic Algorithm for the Mapping Problem

5.2 Population, Chromosomal Representation and Feasibility

GAs population is an array of POP number of individuals. An individual in the population is encoded as an $(N \times M)$-element vector $[X_{1,1}$, $X_{1,2}$, ... $X_{1,M}$, $X_{2,1}$,$X_{2,2}$, ... $X_{2,M}$, ... $X_{N,1}$ $X_{N,2}$, ... $X_{N,M}]$ where N is the number of the servers and M is the number the clients, and it corresponds to a candidate solution to the allocation of servers to clients that provides a candidate solution to the optimization problem. The sub-vector $X_{1,1}$, $X_{1,2}$, ... $X_{1,M}$ corresponds to the clients allocated to server 1, sub-vector $X_{2,1}$, $X_{2,2}$, ... $X_{2,M}$ the clients allocated to server 2, and so forth. An element (gene) $X_{i,j} = 1$ (or 0), for $i \in [1...N]$ and $j \in [1...M]$, indicates the allocation (or deallocation) of server i to customer j.

The initial population of individuals is usually randomly generated. However, in our case, a random generation of individuals introduces too many infeasible individuals to the population. An individual is considered *infeasible*, if it does not satisfy the given constraints (14) and (15). In other words, an individual is infeasible if it is not able to service all the clients because the sum of the active links to each server from a client is greater than the maximum bandwidth of that client, or the sum of the links to each client from a server is greater than the maximum bandwidth of that server. The constraint set (13) is by default satisfied by the above described chromosomal representation of an individual.

In order to reduce the infeasibility problem, the initial individuals generator was written in such a way that it assigns at random one of the servers to a client. Then, it tries to assign another server, if the second server does not make the individual infeasible (too much bandwidth for the client). This method is good in decreasing the infeasibility in the population. It only generates feasible solutions, unless the servers are too slow or overloaded.

5.3 Objective Functions Evaluation and Reproduction Scheme

Using the genes of an individual, the fitness Z of an individual is evaluated as described in (12). Henceforth, the minimal average access time of all clients corresponds to the minimum value of the fitness Z of all feasible individuals.

In GA, the whole population is considered a single reproduction unit within which random selection is performed. Our reproduction scheme involves elitist ranking, followed by random selection of mates from the list of reproduction trials (or copies) assigned to the ranked individuals. In the ranking scheme [2], the individuals are sorted by their fitness value. After sorting, each individual is assigned a rank based on a scale of equidistant values for the population. The ranks assigned to fittest and least-fit individuals are 1.2 and 0.8, respectively. Individuals with ranks greater than 1 are first assigned single copies. Then, the fractional part of their ranks and the ranks of the lower half of individuals are treated as probabilities for random assignment of copies.

It has been found that ranking based selection with a maximum rank of 1.2 produces individual survival percentage of 92 to 98% in different generations[2]. This helps in maintaining population diversity and controlling premature convergence. Elitism is used to exploit good building blocks and to ensure that good candidate solutions are saved if the search is to be truncated at any point. Preservation of the fittest individual is done by replacing the least-fit individual by the fittest-so-far individual if the latter is better than the current-fittest.

5.4 Genetic Operators, Hill-Climbing and Termination Criteria

The genetic operators employed in GA are crossover and mutation. Pairs of individuals are randomly selected from the mating pool. Each pair of these strings undergoes crossover as follows: an integer position k along the individual is selected at random between $[1..(N * M)]$, where $N * M$ is the individual length. The two new individuals are created by swapping all characters(genes) between $k + 1$ and $N * M$ inclusively. In our case, a random selection of the crossover point (i.e. k) produces two infeasible individuals. In order to solve this problem, a smarter mode of crossover was implemented. Rather being totally random, the crossover point is taken only at the end of a client. In this way, we preserve each client's feasibility. Moreover, in order to preserve servers feasibility, many crossover points are tried in order to select a point that produces two feasible individuals. The standard mutation operator is employed. Individuals and gene positions where the alteration of the value is going to occur are selected randomly. A mutation rate of 0.02 and crossover rate of 0.7 [10] are used in our implementation.

To refine the solution quality, a simple problem-specific hill-climbing procedure which may decrease the individual's fitness values is incorporated. Our GA determines how many individuals to "hillclimb" by computing the function *numHill* that starts with tightened hillclimbing rate and increases this rate as the run progresses. We have experimented with different hillclimbing functions and we have found that the following one yields the best solution quality:

$$numHill = \begin{cases} 0.5 \times NumFeasible \, if \; 0 < t \leq 100 \\ 0.7 \times NumFeasible \, if \; t > 100 \end{cases} \tag{16}$$

where t is the current generation and $NumFeasible$ is the number of feasible individuals in the current population. Our GA randomly selects $numHill$ feasible solutions formed in each generation and applies to each of them the following hill-climbing procedure that searches the space nearby an individual solution using a simple "add link" methodology. The procedure randomly selects (N*M)/3 genes and then applies the following to each one: If the value of the gene is 0, then it is provisionally incremented to 1. If the resulting solution is feasible, the provisional change is made permanent and the new solution becomes the incumbent. This means that a link between a certain client and a server is established. This decreases this client's access time, and thus decreases the average access time of the individual. However, if the resulting solution is infeasible, the provisionally incremented gene is restored to its original value. This hill-climbing procedure enables individuals to rapidly climb the peaks, speeding up the evolution process.

The termination criterion is satisfied when we converge to a solution. In this work, convergence is indicated when the best-so-far string does not change its Z value for 15 consecutive generations. We experimented with different values of POP size, and it was found that POP=20 yields the best solution quality.

6 Simulation Study

In order to test the effectiveness of the proposed scheme over a traditional single-server approach, we run rigorous simulation tests estimating how each of the problem parameters reflects on the quality of the delivered service, measured by the AT. It should be noted that although other objective functions could be employed as well, e.g. service costs, client buffer-space, AT variability, etc., these are going to be treated in future extensions of this work.

The parameters that were used in the simulations are listed below:

- N : the number of servers ranged between 2 and 5
- M : the number of clients ranged in $[100, 500 \cdot N]$, targetting a wide variety of loads, including ones that cause server saturation.
- W : the number of installments ranged between 1 and 5
- bw_S : the servers' bandwidth was fixed at $0.08s/MB$ (equiv. to $100Mb/s$)
- bw_C : to represent the variety of clients that could use the service, each client's available bandwidth was randomly selected from 3 possible values:
  - 16 s/MB: equivalent to a 500 kbps ADSL, with a probability of 30%
  - 21 s/MB: equivalent to a 384 kbps ADSL, with a probability of 60%
  - 63 s/MB equivalent to a 128 kbps ISDN, with a probability of 10%
- $bw_{i,j}$: the connection speed between client C_j and server S_i was uniformly selected from $\left[bw_{C_j}, 125s/MB\right]$, the upper limit corresponding to $64kbps$.
- L_j : movie sizes were chosen from a uniform distribution in the $[500, 1000]\,MB$ range, which is typical for MPEG-4-coded feature-length movies.

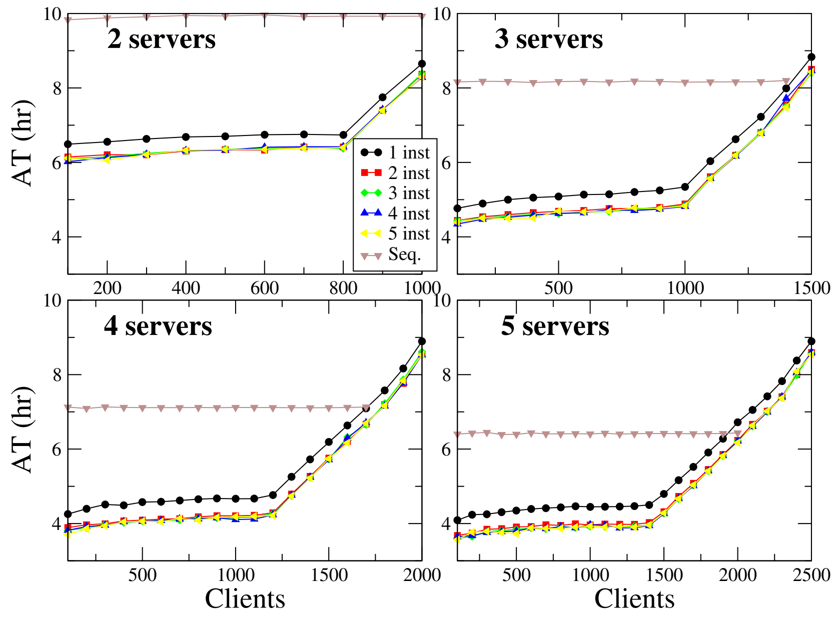

Fig. 3. *AT* vs the number of clients, for different number of servers and installments.

- R_j : playback rates were chosen randomly from a uniform distribution in the $[600, 1200]\,kbps$ range, again typical MPEG-4 rates for high-quality media.

As a benchmark, we calculated the *AT* that the clients would enjoy if they were serviced by a single server only. In this "sequential" approach, each client C_j connects to the fastest available server, i.e. the non-saturated one that exhibits the smallest $bw_{i,j}$.

For each combination of N, M and W values, a total of 100 runs were performed. The average *AT* versus the total number of clients is shown in Figure 3. As can be clearly observed, increasing the number of servers acts favorably in improving the offered service. The shape of the curves is dictated by the availability of server resources. As the number of clients grows, some of them may settle for slow connections, or even for just a single server. As a result, the average *AT* increases but the success of our multi-server GA-based scheme is that the clients can be continuously served even when the sequential scheme begins to fail, as indicated by the absence of data points for high M in the corresponding curves of Figure 3. In these situations a large number of clients is denied service.

It should be noted that the clients are examined in a non-specific order, i.e. they are not sorted prior to pairing them with a server in the sequential scheme. Although a sorting step could possibly extend the viability of the sequential scheme to more clients, it is unlikely that this would improve the exhibited *AT* which is consistently above the one provided by our multi-server scheme.

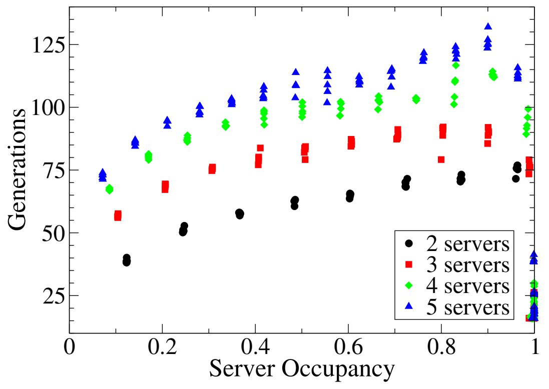

Fig. 4. Generations needed for convergence versus the server occupancy.

Figure 3 also shows that increasing the number of installments improves the average AT. Performance gains are however scant after 2 or 3 installments. More installments might offer an edge for the deployment of an adaptive (responding to connection-state changes) delivery scheme.

The ratio of the bandwidth a server uses for uploading movies over its total bandwidth can be used as a metric of a server's saturation. We refer to this ratio as "server occupancy". Figure 4 shows the number of generations required for the GA to converge, versus the server occupancy. The generations needed take a sharp decline following a slight increase as the server occupancy increases. The reason for this is that the GA fails to improve the original solutions generated, giving up after exceeding the preset threshold for passed generations before improvement. Also, as expected, an increase in the number of servers leads to slightly slower convergence rates due to the expansion of the solution space.

7 Conclusion

In this paper we present a novel approach for optimizing the delivery of movies to multiple clients by multiple servers. We present a genetic algorithm that can generate an optimum or near-optimum mapping of servers to each client request, in an effort to minimize the average AT.

The proposed delivery scheme and mapping algorithm manage to outperform a traditional sequential delivery in every respect, yielding far superior AT while at the same time maintaining normal operation under severe loads that would otherwise cause a denial of service.

Future extension of our work could include:

– Offering different classes of service. Apart from the financial benefits that such an extension would provide, it is only natural that faster and slower clients should be not treated in the same fashion in many different levels.

– Extending the genetic algorithm to handle arbitrary request arrival times. Delivery schedules and mappings could be refined periodically in response to the changing network conditions and/or servers' state (e.g malfunction).

References

1. Awad M., Mansour N., El-Fakih K.: Incremental genetic algorithm. Proc. CSITeA'02, Foz do Iguazu, Brazil, June 6-8 (2002) 24-29.
2. Baker J.E.: Adaptive Selective Methods for Genetic Algorithms. Proc. Int. Conf. on Genetic Algorithms (1985) 101-111.
3. Barlas G., Veeravalli B.: Optimized delivery of Continuous-Media Documents using Distributed Servers. ISCA PDCS 2002 Proceedings, Louisville, USA (2002) 13-19.
4. Boyce J.M., Gaglianello R.: Packet Loss Effects on MPEG Video Sent Over the Public Internet. Proc. ACM Multimedia, Bristol (1998) 181-190.
5. Cai Y., Hua K.A.: An Efficient Bandwidth-Sharing Technique for True Video on Demand Systems. Proc. ACM Multimedia, Orlando (1999) 211-214.
6. Eriksson H.: MBONE: The Multicast Backbone. Comm. of the ACM, Vol. 37, No. 8 (1994) 54-60.
7. Furht B., Westwater R., Ice J.: Multimedia Broadcasting over the Internet: Part I. IEEE Multimedia, October-December (1998) 78-82.
8. Gen M., Cheng R.: Genetic algorithms and engineering optimization. Wiley, New York (2000).
9. Goldberg D.E.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley (1989).
10. Grefenstette J.J.: Optimization of Control Parameters for Genetic Algorithms. IEEE Trans. on Systems, Man, and Cybernetics, Vol. 16, No. 1 (1986) 122-128.
11. Gringeri S., Egorov R., Shuaib K., Lewis A., Basch B.: Robust Compression and Transmission of MPEG-4 Video. Proc. ACM Multimedia, Orlando (1999) 113-120.
12. Jung G.S., Kang K.W., Malluhi Q.: Multithreaded Distributed MPEG-1 Video Delivery in the Internet Environment. Proc. ACM Symp. on Applied Computing, Como, Vol.2 (2000) 592-597.
13. Kuhne G., Kuhmunch C.: Transmitting MPEG-4 Video Streams over the Internet: Problems and Solutions. Proc. ACM Multimedia, Orlando, Vol.2 (1999) 135-138.
14. Lee J.Y.B.: Parallel Video Servers: A tutorial. IEEE Multimedia, Apr-Jun, (1998) 20-28.
15. Miettinen K. et al.: Evolutionary algorithms in engineering and computer science. McGraw-Hill, New York (1999).
16. Pejhan S., Chiang T.H., Zhang Y.Q.: Dynamic Frame rate Control for Video Streams. Proc. ACM Multimedia, Orlando (1999) 141-144.
17. Rejaie R., Handley M.: Quality Adaptation for Congestion Controlled Video Playback over the Internet. Proc. ACM SIGCOMM, Cambridge (1999) 189-200.
18. Rodriguez P., Kirpal A., Biersack E.W.: Parallel-Access for Mirror Sites in the Internet. Proc. of Infocom, Tel-Aviv, Israel (2000).
19. Stockinger H., Samar A., Allcock B., Foster I., Holtman K., Tierney B.: File and Object Replication in Data Grids. J. Cluster Computing, 5(3) (2002) 305-314.
20. Veeravalli B., Barlas G.: Access Time Minimization for Distributed Multimedia Applications. Multimedia Tools & Applications, Kluwer Academic Publishers, Vol. 12 (2000) 235-256.