A Framework for Allocating Clients to Rate-Constrained Multicast Servers

Zongming Fei [a,*] , Mengkun Yang [a] , Mostafa H. Ammar [b] , Ellen W. Zegura [b]

[a] *Department of Computer Science, University of Kentucky Lexington, KY 40506, USA*

[b] *College of Computing, Georgia Tech Atlanta, GA 30332, USA*

Abstract

Using multiple multicast groups is a technique often used to deal with the problem of heterogeneous client capacities by allocating clients with different capacities to different groups. In this paper, we investigate the problem of allocating clients to constrained multicast servers, which, similar to clients, have different capacities. We explore several interesting issues caused by these constraints and propose a framework for generating optimal solutions to the allocation problem. We evaluate the method by simulation and show a substantial performance gain of our algorithm over those considering the server constraints separately.

Key words: Multicast, Replication, Server selection, Rate constraint

1 Introduction

The continuous growth of the Internet population and usage presents us a challenge of providing services scalable to a very large number of clients. Some popular servers can be overwhelmed by such a large volume of client requests that the response time becomes nearly untolerable. Multicast communication has been proposed to achieve scalability by sending data to multiple clients at the same time [1–3]. For example, a server can multicast a live event over the

* Zongming Fei is the corresponding author.
Email addresses: `fei@cs.uky.edu` (Zongming Fei), `myang0@cs.uky.edu` (Mengkun Yang), `ammar@cc.gatech.edu` (Mostafa H. Ammar), `ewz@cc.gatech.edu` (Ellen W. Zegura).

Internet and any client interested can join the multicast group and receive the video and audio streams. Through the aggregation of request processing at the server and the aggregation of bandwidth requirements in the network, a multicast service can reduce the load on the server and network and, therefore, improve the performance perceived by clients.

One well-known problem in multicast communication is the heterogeneity of multicast clients. For example, in a live streaming multicast application, some clients may be capable of receiving at a very high rate while others can only receive at a low rate. The difference can be the result of either the client processing power or the path capacity from the server to it. However, in a session with a single group, clients have to share the fate with each other. The sender may simply transmit at the lowest rate of all clients, or at some fixed rate possibly higher than the lowest rate. In the former case, faster clients have to receive at a rate lower than their capacities. In the latter case, slower clients may experience loss because the server is sending faster than their capacities.

More recently schemes of using multiple multicast groups have been proposed to deal with the problem [4–7]. Instead of forming a single group sending at a certain rate, the server can form multiple multicast groups sending at different rates. Clients with similar capacities can join the same group and the difference among clients within a group is much smaller. Thus the overall performance perceived by clients is improved. The basic assumption of these schemes is that a *single* server forms all these multicast groups and clients are directed to an appropriate group. However, in this paradigm the total rate of all the groups is limited by the capacity of the server, or the bandwidth of the link connecting the server to the network. This problem can be serious if the number of multicast groups becomes larger.

In this paper we want to consider an alternative paradigm (i.e., replication), in which different multicast groups are formed and originated from different servers. Though we somehow relax the single server requirement discussed above, some new problems arise. Prominent among them is the problem of differences among server capacities. This can be caused by either the server itself, or the link connecting the server. For example, one server can be located in a place connected by a DSL line and another server can be located inside the backbone of a major service provider. Another example is that two servers may use machines with different capabilities. These factors put an upper bound on how fast each server can send information out. It is called the *server rate constraint* in this paper. This constraint sometimes changes the nature of the allocation problem and increases the complexity of solutions.

The focus of this paper is on studying the problem of allocating clients to rate-constrained multicast servers. The goal is to optimize the total utility over all clients. We start with the definition of utility functions for clients and the optimization problem of allocating clients to appropriate servers. We then present a framework that can generate optimal solutions for a large category of utility functions. The evaluation shows a substantial performance gain of our algorithm over those considering the constraints on server rates separately.

The rest of this paper is organized as follows. Section 2 formulates the problem and analyzes some implications of server rate constraints. Section 3 proposes a framework for generating optimal solutions and illustrates instantiations of the framework to solve two specific problems. Performance results are presented in Section 4 and related work is discussed in Section 5. Section 6 concludes the paper.

2 Problem Description

In our formulation both servers and clients have rate constraints. Assume we have n clients and the rate (also called *isolated rate* [1]) of client i $(1 \leq i \leq n)$ is r_i. There are m servers and the maximal rate of server j $(1 \leq j \leq m)$ is s_j. Note it is the upper bound of the final sending rate of server j. In the following discussion, we simply call this upper bound *server rate* and its final sending rate *group rate*. We can assume that $r_i \leq r_{i+1}$ $(1 \leq i \leq n-1)$ and $s_j \leq s_{j+1}$ $(1 \leq j \leq m-1)$.

An allocation of clients to servers can be represented by a function $\phi : [1..n] \rightarrow [1..m]$ and $\phi(i) = j$ indicates that client i is allocated to server j. A utility function u is defined to represent the performance of a client. It can be the actual receiving rate, the satisfaction level, relative fairness, or some other measures showing what is observed by the client. If a client has isolated rate r and is allocated to a multicast group operating at rate x, the utility of this client will be $u(r, x)$. One client may have a utility function different from another client. The value of most utility functions increases monotonically as x increases when $x < r$, decreases monotonically when $x > r$, and has the maximal value at $x = r$. The optimal allocation is the one that maximizes the sum of utilities over all clients. If the utility function satisfies the monotonicity property, the optimal allocation has the property that $i_1 < i_2$ implies $\phi(i_1) \leq \phi(i_2)$. In the rest of this paper we only consider those allocations with this property.

We use x_j to represent the group rate of server j and obviously $0 \leq x_j \leq s_j$.

[1] As defined in [8], it can be the rate that would be achieved if this were the only client in the multicast group, or the minimum of the max-min fair shares in the links from servers to the client.

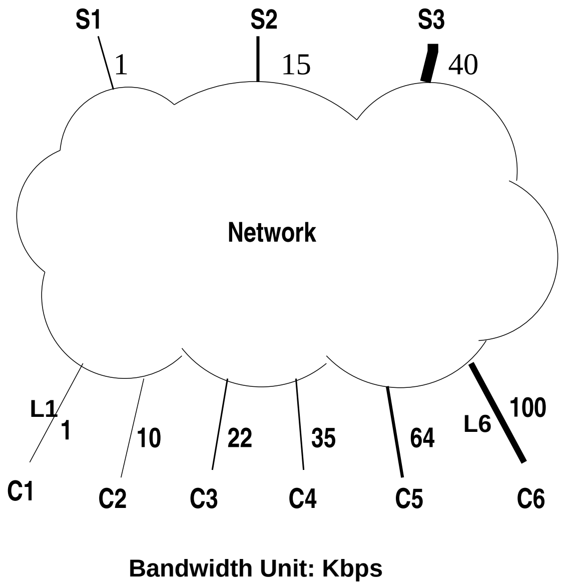

Fig. 1. An example of allocation problem

The goal of maximizing the sum of utilities over all clients is to maximize

$$\sum_{i=1}^{n} u_i(r_i, x_{\phi(i)}) \tag{1}$$

subject to $0 \le x_j \le s_j$ $(1 \le j \le m)$, where u_i is the utility function of client i $(1 \le i \le n)$. The allocation problem is to determine function ϕ, which defines how clients are allocated to servers, and group rates x_j's $(1 \le j \le m)$, which determine the actual receiving rates of clients.

For example, in Fig. 1 we have six clients ($C1$ to $C6$) and their isolated rates are 1, 10, 22, 35, 64, 100, respectively. Three servers ($S1$ to $S3$) have rate constraints 1, 15, 40, respectively. The final receiving rate of a client depends on which server it is allocated to and what the group rate of that server is.

Considering the property that $i_1 < i_2$ implies $\phi(i_1) \le \phi(i_2)$, we observe that the allocation problem can be transformed into selecting $m-1$ indices from n possible locations. If we define $k_1 = 1$ and $k_{m+1} = n+1$, the problem is to select $m-1$ indices k_j $(2 \le j \le m$ and $1 \le k_j \le k_{j+1} \le n)$, such that client i with $k_j \le i < k_{j+1}$ is allocated to server j. In Fig. 2, we have 11 clients with indices from 1 to 11, and three servers ($n = 11$ and $m = 3$). Thus $k_1 = 1$ and $k_4 = 12$. If we select $m-1$ indices as follows: $k_2 = 4$ and $k_3 = 9$, then clients 1,2,3 will be grouped together and allocated to server 1, clients 4,5,6,7,8 to server 2, and clients 9,10,11 to server 3.

Server rate constraints capture the difference in capacities of servers, similar to the different rates of clients. The problem with server rate constraints is a generalization of the problem without them. If we let $s_j = \infty$ $(1 \le j \le m)$

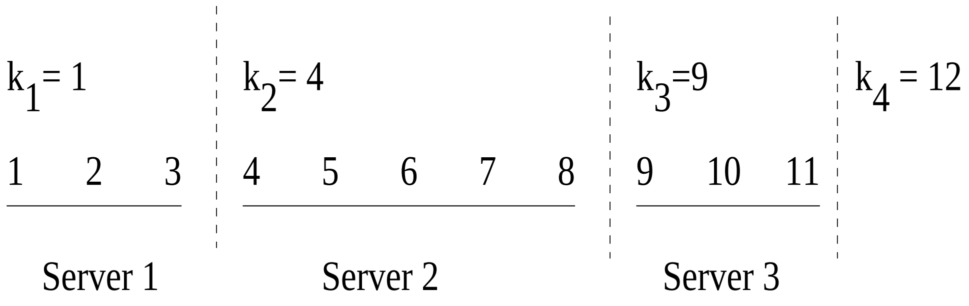

Fig. 2. Relationship between allocation and selecting indices

in the above formulation, we instantiate it to the allocation problem without server rate constraints.

The constraints on server rates increase the complexity of the problem. During the optimization process, we have to take server rate constraints into consideration in addition to client rates. One minor implication is that the problem of dividing clients into m groups becomes the problem of dividing clients into *at most* m groups. It is easy to understand that when we have enough number of servers with high rates, we do not want to allocate clients to those servers with a very low rate. Even if the server rates are not too low, there are still cases in which the optimal solution has less than m groups. For example, suppose four servers (s_1, s_2, s_3 and s_4) have rates $3, 6, 7$ and 8, respectively, five clients (c_1, c_2, c_3, c_4 and c_5) have rates $3, 5, 9, 11$ and 13, respectively. We require that all clients be in some groups and the group rate be the lowest of the server rate and the rates of clients allocated to this server. Then one optimal solution is to divide five clients into three groups instead of four, with client c_1 allocated to s_1, client c_2 to s_2, and clients c_3, c_4, c_5 to s_4. Note no client is allocated to server s_3.

We now define two specific allocation problems based on the definitions of utility functions. We will use them to evaluate the performance of the solution presented in the next section.

- *Max Rate Problem.* In this problem, we define $u_i(r, x) = x$ if $x \leq r$ and $u_i(r, x) = 0$ if $x > r$, for all $1 \leq i \leq n$. The goal becomes maximizing the total rate of clients. Here we assume that clients cannot tolerate any loss and it is of no use at all if the group rate is higher than the rate of the client. We also require that all clients be in some groups and the group rate be set to the minimum of the server rate and the rates of the clients allocated to it. We have $\sum_{i=1}^{n} u(r_i, x_{\phi(i)}) = \sum_{i=1}^{n} x_{\phi(i)}$, where $x_{\phi(i)} \leq r_i$ and $x_{\phi(i)} \leq s_{\phi(i)}$. The optimization can also be reduced to maximizing $\sum_{j=1}^{m} min(r_{k_j}, s_j)(k_{j+1} - k_j)$.
- *Max Satisfaction Problem.* In this problem, we define the utility function $u_i(r, x) = u(r, x) = \frac{min(r,x)}{max(r,x)}$, which is a linear function ($= x/r$) of x in the range $[0..r]$ and an inverse function ($= r/x$) of x in the range $[r..\infty]$. Here we assume that clients can tolerate loss and we may set the group rate of

a server higher than the isolated rates of some clients to increase the sum of utilities of the clients in the group. The utility function indicates the satisfaction level of each individual client. We want to maximize the overall satisfaction level in this problem.

3 A Framework for Generating Optimal Allocations

The brute force solution to the problem involves trying every possible way to select $m-1$ indices from n possible positions. We develop a framework that can solve the allocation problem in polynomial time. We first transform the original problem to the shortest path problem and then use Dijkstra's algorithm [9] to solve it. We present instantiations of the framework to get algorithms for solving the *max rate problem* and the *max satisfaction problem*.

Informally, we represent possible grouping of clients and their association with a server by a directed graph, with the vertices arranged in rows and columns as follows. For the problem of allocating n clients to m servers, we have $n \times m+1$ vertices. Fig. 3 illustrates the graph generated for the problem which has six clients and three servers as shown in Fig. 1. There is one column of vertices for each server; there is one row of vertices for each client. Vertex $v_{i,j}$ in row i and column j represents the potential allocation that clients selecting server j will start from client i. An edge from $v_{i,j}$ to $v_{k,j+1}$ $(i \leq k)$ represents that clients from i to $k-1$ will be grouped together and associated with server j.

The weight on an edge describes the performance associated with the choice. We use the weight on the edge to represent the "deviation" from the ideal value for clients i to $k-1$ because of their association with server j. For example, in the max rate problem the ideal total rate of n clients is $\sum_{i=1}^{n} r_i$. Instead of maximizing the actual total rate of clients directly, our algorithm solves it by minimizing the deviation from the above ideal rate. For the edge from $v_{i,j}$ to $v_{k,j+1}$, all except client i will therefore receive at a rate lower than their isolated rates and the deviation contributed by this edge is $\sum_{\ell=i}^{k-1}(r_\ell - min(r_i, s_j))$.

A solution is represented as a path including one vertex per column in the graph. With the edges and weights as described, the shortest path corresponds to an optimal solution.

Formally, we define a deficit digraph $G_d = (V_d, E_d)$, where $V_d = \{v_{i,j}\} \cup \{v_{n+1,m+1}\}$ $(i = 1, \cdots, n, j = 1, \cdots, m)$ and E_d is given as follows:

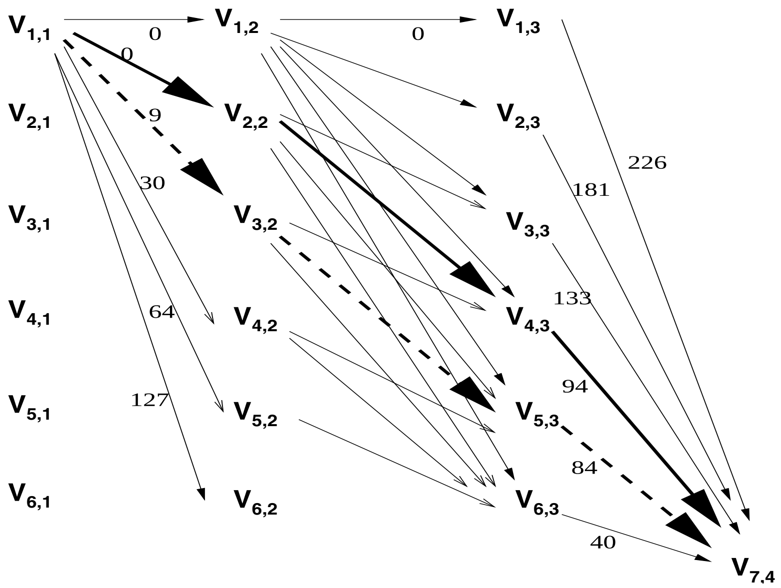

Fig. 3. An example solution

1) For $1 \le i \le n$, connect $v_{1,1}$ to $v_{i,2}$;

2) For $1 \le i < k \le n,\ \ 2 \le j \le m-1$, connect $v_{i,j}$ to $v_{k,j+1}$;

3) For $1 \le i \le n$, connect $v_{i,m}$ to $v_{n+1,m+1}$;

4) For $2 \le j \le m-1$, connect $v_{1,j}$ to $v_{1,j+1}$.

The weight on the edge from $v_{i,j}$ to $v_{k,j+1}$ is denoted by $D_C(i,j,k)$ and represents the deviation from the ideal value if clients i to $k-1$ are allocated to server j. Define $D_C(i,j,k) = 0$ if $i \ge k$. Using Dijkstra's algorithm, we can compute the shortest path from $v_{1,1}$ to $v_{n+1,m+1}$. The path must be in the form $(v_{1,1}, v_{k_2,2}, \cdots, v_{k_j,j}, \cdots, v_{k_m,m}, v_{n+1,m+1})$. These k_i's $(2 \le i \le m)$ are where we want to partition the client set.

The remaining problem is how to calculate $D_C(i,j,k)$. The complexity depends on the definition of utility functions $u_i(r,x)$. We show solutions to the two problems defined in Section 2.

- Max rate problem. The group rate x_j of server j is the minimum of the server rate and the lowest client rate, i.e., $x_j = min(r_i, s_j)$. The deficit caused by client ℓ $(i \le \ell \le k-1)$ is $r_\ell - x_j = r_\ell - min(r_i, s_j)$. Therefore, the total deviation is $D_C(i,j,k) = \sum_{\ell=i}^{k-1}(r_\ell - min(r_i, s_j))$. The total rate of clients in group j is $\sum_{\ell=i}^{k-1} min(r_i, s_j)$. The maximum possible total rate of clients in group j is $\sum_{\ell=i}^{k-1} r_\ell$. Therefore $D_C(i,j,k)$ is the relative difference of the total rate achieved by clients in the group from the maximum total rate (i.e., the ideal rate).

  If we do not consider server rate constraints (or assume all s_j's are fairly large), the shortest path in Fig. 3 is $(v_{1,1}, v_{3,2}, v_{5,3}, v_{7,4})$, shown as dotted bold edges, and the optimal solution to the problem is to set servers s_1, s_2, s_3 at rates r_1, r_3, r_5, or 1, 22, 64 and allocate clients c_1, c_2 to s_1, clients c_3, c_4 to s_2 and clients c_5, c_6 to s_3. If we do consider server rate constraints and assume that $s_1 = 1, s_2 = 15, s_3 = 40$ as in Fig. 1. The weights in Fig. 3 are calculated for this case. The shortest path is $(v_{1,1}, v_{2,2}, v_{4,3}, v_{7,4})$, shown as solid bold edges. The optimal solution is to allocate client c_1 to server s_1, clients c_2, c_3 to s_2, and clients c_4, c_5, c_6 to s_3.

- Max satisfaction problem. The calculation of $D_C(i, j, k)$ is more complicated because the group rate can be larger than the lowest rate of clients. For the group of clients from i to $k - 1$, the ideal satisfaction value is $k - i$, i.e., each client has a satisfaction level equal to 1. Because they are allocated to server j, the penalty is the deviation from the ideal value. If we can solve the optimization problem of maximizing the satisfaction level $\sum_i^{k-1} u(r_i, x_j)$ of this group by choosing an appropriate group rate x_j, we can get the deviation $D_C(i, j, k) = k - i - max_{x_j} \sum_i^{k-1} u(r_i, x_j)$. Note *max* in the formula means maximize over x_j.

  According to [8], for the utility functions satisfying the condition that $u(r, x)$ has one point of discontinuity at $x = r$ and the second derivative greater than or equal to 0 in other x values, $\sum_i^{k-1} u(r_i, x)$ will be maximized when x is one of r_ℓ's $(i \le \ell \le k - 1)$ or s_j. For this category of utility functions, the algorithm only needs to test these r_ℓ's and s_j. Assuming that the goal is maximized at $x = a$, we get $D_C(i, j, k) = k - i - \sum_i^{k-1} u(r_i, a)$.

Our framework can be applied to a wide range of problems, as long as we can find a way to calculate the minimal deviation $D_C(i, j, k)$. For example, when clients have different utility functions, we can use their respective utility functions in the calculation of weight $D_C(i, j, k)$ and get an optimal solution to the problem. We can even use the framework to minimize the total load on all the servers by defining an appropriate $D_C(i, j, k)$ function.

The correctness of the framework is straightforward. Any allocation can be mapped to a path in the directed graph, with the length of the path being the minimum possible deviation from the ideal value by this allocation. The fact that the path found by our algorithm is the shortest implies that the solution has a minimal deviation from the ideal value. Therefore, it is an optimal solution.

The complexity of the algorithms consists of two parts, the complexity of computing the shortest path using Dijkstra's algorithm and the complexity of computing the edge weights. The complexity of Dijkstra's algorithm with a simple implementation is $O(V^2)$, where V is the number of vertices in the graph. We have $O(mn)$ vertices and the complexity is $O(m^2n^2)$. The complexity of computing the edge weights depends on the allocation problem. For the max rate problem, each edge needs to add the rates of at most n clients together and the total number of edges is $O(mn^2)$. So the complexity of calculating the edge weights is $O(mn^3)$. For the max satisfaction problem, we need to test each r_i in the group and s_j to see whether we can get the maximum satisfaction level for the group. For each edge the time is $O(n^2)$. Therefore the complexity in this case is $O(mn^4)$.

One thing worth mentioning is the way the framework deals with those optimal solutions with less than m groups. We intentionally avoid those edges from $v_{i,j}$ to $v_{i,j+1}$ for $2 \leq i \leq n,\ \ 2 \leq j \leq m-1$, even though these edges can be in the paths corresponding to optimal solutions with less than m groups. For example, the path $(v_{1,1}, v_{4,2}, v_{4,3}, v_{7,4})$ corresponds to a solution with two groups, one group consisting of clients 1, 2, 3 and the other group consisting of clients 4, 5, 6. The difficulty with this path lies in calculating the weight of edge $\langle v_{1,1}, v_{4,2}\rangle$. Edge $\langle v_{4,2}, v_{4,3}\rangle$ in the path means that there are no clients associated with server 2. Therefore we should use the rate of server 2 instead of the rate of server 1 to calculate the weight of edge $\langle v_{1,1}, v_{4,2}\rangle$. However, the edge weight calculated this way cannot be used in other pathes where there are clients associated with both server 2 and server 3. The framework solves the problem by eliminating edges such as $\langle v_{4,2}, v_{4,3}\rangle$. Instead we add edges from $v_{1,j}$ to $v_{1,j+1}$ $(1 \leq j \leq m-1)$. Solutions with less than m groups can be represented as a path with an edge from $v_{1,1}$ to $v_{1,2}$ (and possibly other edges from $v_{1,j}$ to $v_{1,j+1}$, where $2 \leq j \leq m-1$). If any of these solutions is optimal, it will be found by the framework. For example, path $(v_{1,1}, v_{1,2}, v_{4,3}, v_{7,4})$ represents the same allocation as path $(v_{1,1}, v_{4,2}, v_{4,3}, v_{7,4})$ and can be found by the algorithm. In summary the framework accommodates optimal solutions with less than m groups nicely and eliminates the problem of calculating the weights of edges such as $\langle v_{1,1}, v_{4,3}\rangle$ when edges $\langle v_{i,j}, v_{i,j+1}\rangle (i \geq 2)$ are allowed.

3.1 Implementation Considerations

Two problems need to be solved in implementing the framework in a practical network environment. The first problem is the determination of isolated rates of clients and the constraints on the rates of servers. The second problem is where and how the algorithm will be implemented.

The determination of the rates of clients can be performed at each client node. After getting a list of servers, a client can probe the network to estimate the bandwidth from each server by using one of several existing methods [10–12]. The maximum among them is taken as the isolated rate of the client. In most cases, this is also the rate of the access link to the network from this client. The determination of the constraints on server rates cannot use the same probing technique because it is hard to determine the destination to probe. A reasonable alternative is to depend on more static information about a server.

Usually when we set up a server, we know the bandwidth of the access link and the maximum sending rate of the server. The minimum of these two values is taken as the constraint on the rate of this server.

We can assume that these replicated servers are established by a replication service provider. An application layer anycasting architecture [13] can be used to allocate clients to one of these replicated multicast servers. For each set of replicated multicast servers, there is an authoritative resolver associated with it. There is also a resolver in each domain where clients are located. These resolvers form a DNS-like hierarchy to direct clients to an appropriate multicast server. When a client sends a request to a local anycast resolver, it will include its isolated rate in the request. These anycast resolvers collect the rate information and send it to the authoritative resolver for this replication service. On the other hand, all multicast servers are supposed to exchange their constraint information with the authoritative resolver. Because it has information about the rates of all the clients and the constraints on the rates of all the servers, the authoritative resolver is a natural place for the algorithm proposed in this paper to be executed. The results will be used to direct the server to set an appropriate rate, and distributed to other non-authoritative resolvers to direct each client to an appropriate server. Note for each service consisting of a set of replicated servers, there is an authoritative resolver where the algorithm is executed. However we will have a different authoritative resolver for a different set of replicated multicast servers. Therefore the algorithm will be executed at different places over the network for different sets of replicated servers.

4 Performance Evaluation

We evaluate the performance of the framework for generating optimal solutions to the allocation problem. Specifically we study the performance of the derived algorithms for solving the two problems defined in Section 2.

In the evaluation, we use 400 clients and vary their rates from 0 to 1. The distribution of these rates is uniform, exponential, or normal. The mean of rates is always set to 0.5. For the normal distribution, the variance is set to 1. We only use values in the range (0..1) in the cases of exponential and normal distributions. We experiment with 1 to 10 servers and their rates are randomly selected from (0..1), following uniform distribution. For each data point in the results, we test 1000 times and get an average value.

In the max rate problem, the performance measure is the total rate. For a specific allocation, we set the group rate of a server to the minimum of the server rate and client rates. We calculate the total rate of all clients. This rate is normalized by using the ratio of this rate over the sum of isolated rates of

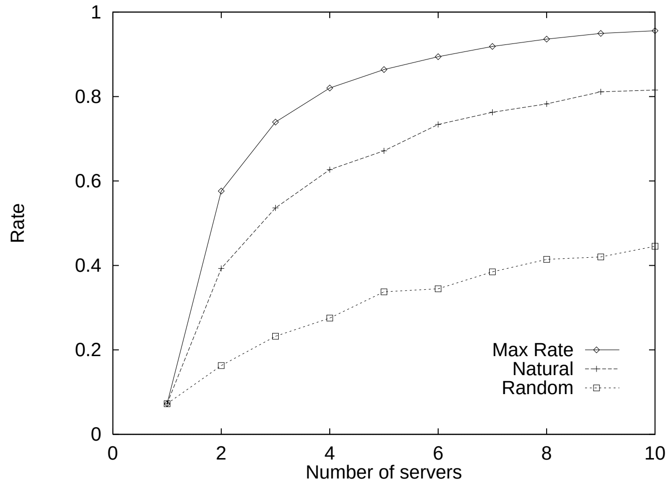

Fig. 4. The rate (uniform distribution of client rates)

clients.

In the max satisfaction problem, the performance measure is the total satisfaction level. For a specific allocation, the group rate of a server is either the server rate, or one of the client rates that are less than the server rate, whichever leads to a higher total satisfaction level wins. We calculate the utility for each client and use the average value over all clients as the total satisfaction level.

In each problem, we compare the optimal algorithm with two other algorithms.

- Random selection. Each client randomly selects a server.
- Natural selection. For each client, let S be the set of servers whose rates are lower than or equal to the rate of this client; then from set S, select the server with the highest rate for this client. Or simply, each client selects a server with the highest rate from those having a lower (or equal) rate than this client.

These two algorithms have a lower complexity than the algorithm proposed in the previous section. Assume we have n clients and m servers. For the random selection algorithm, we only need to go through each client and generate a random number in $[1..n]$. Thus the complexity of the random algorithm is $O(n)$. In the natural selection algorithm, for each client, we need to compare its rate with the rates of m servers and select an appropriate server. The simplest implementation results in the complexity of $O(mn)$. Using the binary search, the complexity can be reduced to $O(n \log m)$. Note that these two algorithms do not need to calculate the weights as in the optimal algorithm.

Fig. 4 and 5 show the results when the rates of clients are uniformly distributed over $(0..1)$. In Fig. 4, we compare the performance of the three algorithms used to solve the *max rate problem*. The average rate of clients increases as the number of servers increases from 1 to 10. As expected, the random algo-

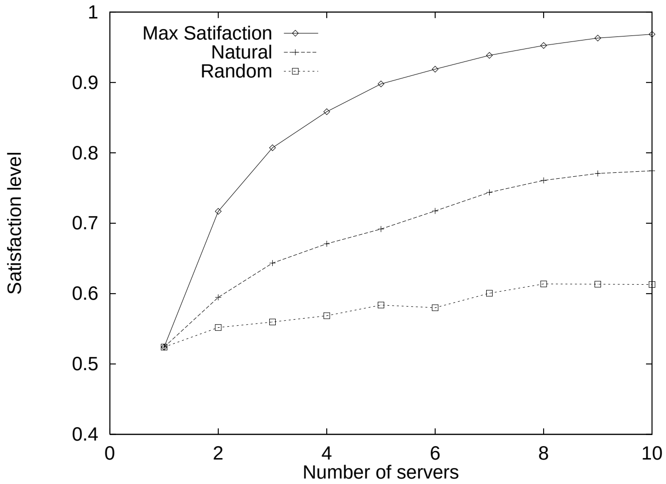

Fig. 5. The satisfaction level (uniform distribution of client rates)

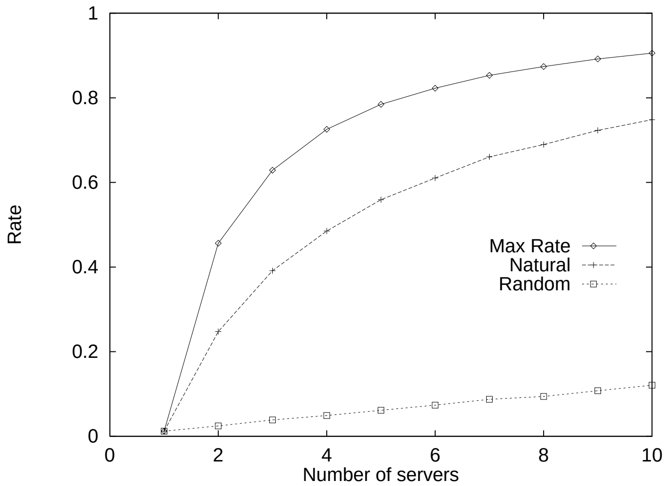

Fig. 6. The rate (exponential distribution of client rates)

rithm performs worst and the max rate algorithm performs best. We notice our algorithm has a significant gain over the *natural selection* algorithm. It demonstrates a sharp increase when the number of servers increases from 1 to 6. Beyond that, the curve becomes flatter. This implies that the performance benefits will be obvious as we add replicas, but only up to a small number of them.

Fig. 5 shows the results when the algorithms are used to solve the *max satisfaction problem*. The algorithm instantiated from our framework leads to the largest satisfaction level, compared to the natural selection and the random selection algorithms. The difference between our algorithm and the natural selection algorithm is larger than in the max rate problem. This is partly because the group rate of a server can be higher than the lowest rate of clients in this case and the solution space is larger (more flexibility). Thus the difference between the optimal solution and non-optimal solutions becomes larger.

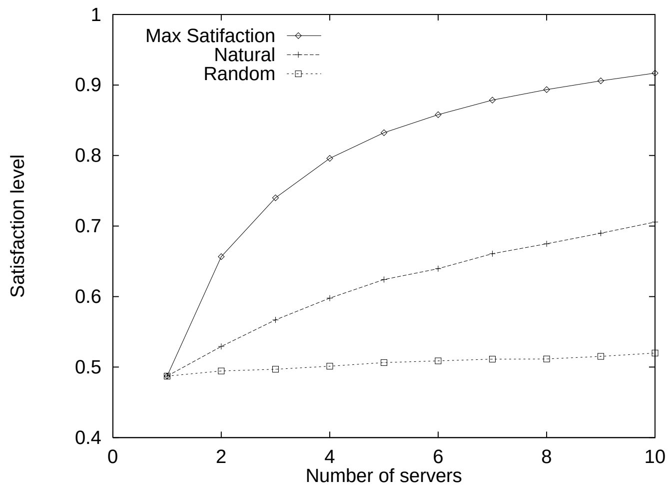

Fig. 7. The satisfaction level (exponential distribution of client rates)

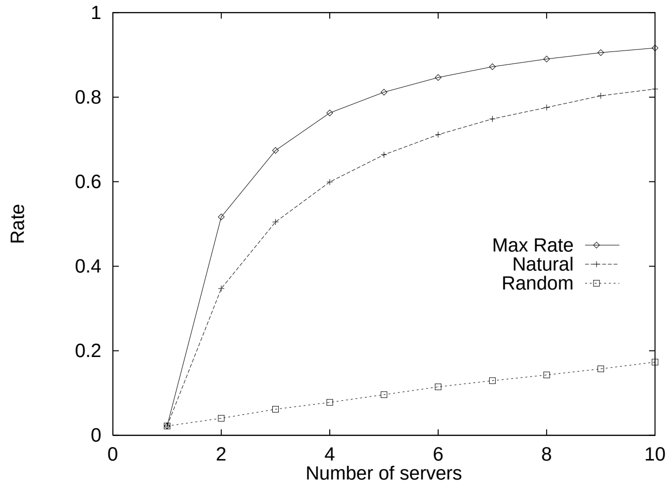

Fig. 8. The rate (normal distribution of client rates)

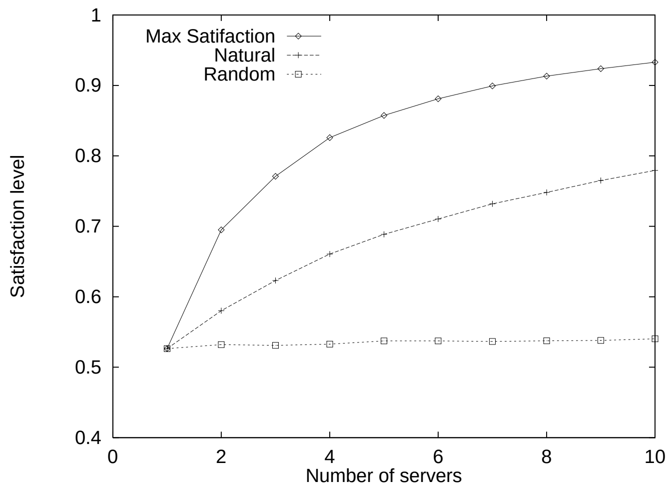

Fig. 9. The satisfaction level (normal distribution of client rates)

Fig. 6 to 9 show the results when the distribution of client rates is exponential or normal. The basic patterns are similar to the case of uniform distribution. Both the rate and the satisfaction level are a little bit smaller, possibly because these two distributions are more skewed, while server rates are uniformly distributed over the range.

We also compare the optimal solution to the algorithm (called *separate algorithm*) that considers server rate constraints separately [14]. More specifically, it is a modification of the optimal solution. First it divides clients into groups using the optimal algorithm, assuming that there are no server rate constraints. Then it associates each group with a server. Finally it recalculates the total rate and the satisfaction level as stated at the beginning of this section. We compare the performance of the optimal solutions with the separate algorithm. They start with the same value when there is only one server and the difference between them becomes larger when the number of servers increases. This tells us that when there are server rate constraints, we have to integrate them into the solution process to get the best result possible.

5 Related Work

Server replication and selection has been extensively studied in the unicast environment [15,16,11,13,17,18]. The emphasis can be on the architectural framework for providing selection services [19,20,18], or selection strategies and methods for improving performance [21,11], or specific issues in a particular kind of networks (e.g., differentiated service network) [17].

The use of multiple multicast groups or channels has been proposed to deal with the problem of heterogeneous client capacities. Ammar et al proposed the idea of destination set splitting [4]. The clients are divided into groups based on their capabilities and the sender sends to different groups at different rates controlled by the feedback from clients in each group. Bhattacharyya et al designed a scheme to minimize the transfer time by using multiple multicast groups in bulk data transfer [5]. Each client may join multiple channels and is characterized by a "rate" that represents how fast it can receive data. A simple rate control scheme was presented to determine the rate of each multicast channel. Multiple multicast channels were also used in a reliable multicast scheme to reduce the end-host processing cost and network bandwidth usage [6]. The basic assumption of these schemes is that the rate of a group is mainly limited by the capacities of clients. In contrast, our work generalizes the problem to the case in which servers also have different capacities and studies its impact on the decision of directing clients to an appropriate server.

6 Concluding Remarks

Heterogeneity is one of the most important problems in multicast communication. In this paper, we investigate the issue of difference in server capacities and how it affects the allocation of clients. The contributions of the paper can be summarized in three aspects. First we define the optimization problem of allocating clients with different capacities to servers, which also have rate constraints. We analyze the implications of this added complexity. Second we give a general framework for solving the optimization problem efficiently. It has been demonstrated that the conceptual framework can be easily instantiated to solve two specific problems. Third the performance evaluation illustrates the potential benefits of our proposed algorithms over the random and the natural selection algorithms. We also demonstrate the advantages of our algorithms over those considering server rate constraints separately in the solution process.

Our future work includes developing protocols to use the algorithms in multicast applications and services. The solution given in the paper assumes global knowledge about clients and their utility functions and does not consider dynamic behavior of clients joining or leaving the multicast session. We will investigate heuristic solutions to these problems.

7 Acknowledgment

The work of Zongming Fei and Mengkun Yang was supported in part by the National Science Foundation under Grants CCR-0204304 and EIA-0101242. The work of Mostafa H. Ammar and Ellen W. Zegura was supported in part by the National Science Foundation under Grant ANI-9973115. We also want to thank anonymous reviewers for their valuable comments that help improve the paper.

References

[1] C. Diot, W. Dabbous, J. Crowcroft, Multipoint communication: A survey of protocols, functions and mechanisms, IEEE Journal on Selected Areas in Communications 15 (1997) 277–290.

[2] Rodriguez, E. Biersack, Continuous multicast push of web documents over the Internet, IEEE Network Magazine 12 (2) (1998) 18–31.

[3] A. Dan, D. Sitaram, P. Shahabuddin, Scheduling policies for an on-demand video server with batching, in: Proceedings of ACM Multimedia 94, 1994, pp. 15–23.

[4] M. H. Ammar, L.-R. Wu, Improving the throughput of point-to-multipoint arq protocols through destination set splitting, in: Proceedings of INFOCOM'92, 1992, pp. 262–71.

[5] S. Bhattacharyya, J. Kurose, D. Towsley, R. Nagarajan, Efficient rate-controlled bulk data transfer using multiple multicast groups, in: Proceedings of IEEE INFOCOM'98, 1998, pp. 1172–1179, San Francisco, CA.

[6] S. Kasera, G. Hjalmtysson, D. Towsley, J. Kurose, Scalable reliable multicast using multiple multicast channels, IEEE/ACM Transactions on Networking 8 (2000) 294–310.

[7] D. Rubenstein, J. Kurose, D. Towsley, The impact of multicast layering on network fairness, in: Proceedings of ACM SIGCOMM'99, 1999, Cambridge, MA.

[8] T. Jiang, M. H. Ammar, E. W. Zegura, Inter-receiver fairness: A novel performance measure for multicast ABR sessions, in: Proceedings of ACM SIGMETRICS 98/Performance 98, 1998, pp. 202–211, Madison, Wisconsin.

[9] E. W. Dijkstra, A note on two problems in connexion with graphs, Numerische Mathematik 1 (1959) 269–271.

[10] M. Jain, C. Dovrolis, End-to-end available bandwidth: Measurement methodology, dynamics, and relation with tcp throughput, in: Proceedings of SIGCOMM 2002, 2002, pp. 295–308, Pittsburgh, PA.

[11] R. L. Carter, M. E. Crovella, Server selection using dynamic path characterization in wide-area networks, in: Proceedings of INFOCOM'97, 1997, pp. 1014–1021, Kobe, Japan.

[12] S. Keshav, A control-theoretic approach to flow control, in: Proceedings of SIGCOMM 91, Sept., 1991.

[13] E. Zegura, M. Ammar, Z. Fei, S. Bhattacharjee, Application-layer anycasting: A server selection architecture and use in a replicated web service, IEEE/ACM Transactions on Networking 8 (2000) 455–466.

[14] Z. Fei, M. Yang, M. Ammar, E. Zegura, Allocating clients to constrained multicast servers: an optimal solution, in: Proceedings of 10th IEEE International Conference on Computer Communications and Networks, 2001, Scottsdale, AZ.

[15] A. Myers, P. Dinda, H. Zhang, Performance characteristics of mirror servers on the Internet, in: Proceedings of INFOCOM'99, 1999, pp. 304–312, New York.

[16] S. Seshan, M. Stemm, R. H. Katz, SPAND: Shared passive network performance discovery, in: Proc 1st Usenix Symposium on Internet Technologies and Systems (USITS '97), 1997, Monterey, CA.

[17] F. Hao, E. W. Zegura, M. H. Ammar, Supporting server selection in differentiated service networks, in: Proceedings of IEEE Infocom'2001, 2001, pp. 659–668, Anchorage, Alaska.

[18] D. Katabi, J. Wroclawski, A framework for global IP-anycast (GIA), in: Proceedings of ACM SIGCOMM'2000, 2000, pp. 1–13.

[19] C. Partridge, T. Mendez, W. Milliken, Host anycasting service, RFC 1546 .

[20] S. Bhattacharjee, M. H. Ammar, E. W. Zegura, V. Shah, Z. Fei, Application layer anycasting, in: Proceedings of INFOCOM'97, 1997, pp. 1388–1396, Kobe, Japan.

[21] Z. Fei, S. Bhattacharjee, E. Zegura, M. Ammar, A novel server selection technique for improving the response time of a replicated service, in: Proceedings of INFOCOM'98, 1998, pp. 783–791, San Francisco, CA.