Efficient Server Replication and Client Re-direction for Multicast Services

Zongming Fei[a] and Mostafa Ammar[b] and Ellen Zegura[b]

[a]Department of Computer Science, University of Kentucky,
773 Anderson Hall, Lexington, KY 40506-0046
[b]Networking and Telecommunications Group, College of Computing,
Georgia Tech, Atlanta, GA 30332-0280

ABSTRACT

Server replication and multicasting are well-established techniques for increasing the capacity of a networked service and improving client performance. In this paper, we consider the combination of these two techniques. Specifically, we investigate the problem of selecting amongst *rate-adaptive multicast servers*, which adjust their sending rate based on network conditions and/or feedback from clients. Effective server rate adaptation can lead to efficient utilization of network resources and performance improvement perceived by clients. In this study of adaptive multicast server selection, we explore some fundamental issues and investigate the implications of different selection strategies on the performance perceived by clients. We first formulate several optimization problems based on different performance measures. We prove that the general problem is NP-hard and then present a special case with an optimal polynomial-time solution. We then design several heuristics for the general problem. We investigate the performance of the heuristics through simulation and show that our heuristics can improve the performance perceived by clients over other strategies.

Keywords: Multicast, Server Selection, Replication, Adaptive Server

1. INTRODUCTION

With the explosive growth of the Internet, networked services must potentially handle a large number of clients. An increase in service demand often leads to system performance implications: degraded performance as perceived by the client, possible rejection of the request due to server overload, and poor performance observed by other network connections due to hot spots.

To help maintain performance under higher demands, there have been several proposals for multicast services in which a server delivers information to multiple clients simultaneously.[1–3] 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 the clients.

Multicast servers may be replicated for several reasons: 1) To distribute the load among *on-demand* multicast servers, those receive explicit requests from clients and aggregate them according to some criteria to decide which data objects should be multicast.[3] 2) To distribute the "feedback load" on the servers or entities along the multicast tree from the servers. Examples of this feedback include acknowledgements required in many reliable multicast protocols[4] or loss information required for adaptive video transmission.[5] 3) To improve quality of service provided to clients and reduce the delivery cost. Replication of multicast servers provides clients the opportunity to join different multicast trees. From a client's perspective, joining a particular routing tree may be preferable because that tree may give better quality of service. For example servers that deliver data over a shorter (or lower delay) multicast routing path may be preferred. From a global perspective, having a client join a particular tree may reduce the overall cost of providing the service. For example the overall cost of providing a service may be reduced if a client is made to receive data from a server which is at the root of the closest *tree*; i.e., the tree requiring the shortest graft to reach the client.

Multicast server replication, however, needs to be coupled with an appropriate server selection mechanism.[6] In this paper, we are specifically interested in the problem of selecting amongst *rate-adaptive multicast servers*, which adjust their sending rate based on network conditions and/or feedback from clients.

This work has been supported in part by NSF grant ANI-9973115 and by research funding from Sprint.

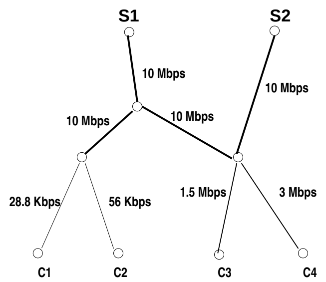


Figure 1. An example

One of the key issues in a multicast service is how fast a server should transmit. There are several possible ways that a server rate can be determined. A simple and often-used method is that the sending rate is pre-determined. The problem with this method is obvious. If the rate is set too low, either the quality of streams sent out is poor (in the case of a video multicast service), or it takes a long time to complete a task (in the case of bulk data transfer). If the rate is too high, other problems can occur. In a heterogeneous environment, the bandwidth of some links from the server to clients may be lower than the server rate. Packets can be lost due to the rate mismatch, and links can be congested. The reception capacities of clients can also be very different from one another. Some fast clients may be capable of processing the packets, while other slower clients cannot.

An alternative to pre-determined rates is that the server rate is adjusted based on feedback from clients. Effective rate adaptation of multicast servers can lead to efficient utilization of network resources and performance improvement perceived by clients. More importantly, it can avoid congesting low capacity links in the network. A simple policy is to set the server rate to the minimum share of bandwidth on all links in the multicast tree. In this paper, we consider this adjustment scheme.

The problem with an adaptive multicast server is that the server rate will be very low if there is even a single low bandwidth link in the multicast tree. For example, as shown in Figure 1, clients $C1$ to $C4$ connect to the network with different capacity access links, ranging from 28.8 Kbps to 3 Mbps. If there is only one multicast server (e.g., $S1$), the server will adapt to sending at the rate of the slowest link, i.e., 28.8 Kbps. This jeopardizes the performance of those clients that can potentially receive at a higher rate. Multicast server replication can be introduced to deal with this problem. If we have multiple servers (e.g., $S1, S2$), we can allocate clients with different capacities to different servers, and they can receive at a higher rate. For example, if we allocate clients $C1$ and $C2$ to server $S1$ and clients $C3$ and $C4$ to server $S2$, then the rate of $C3$ and $C4$ can be increased to 1.5 Mbps.

It is obvious from the above example that an appropriate server selection mechanism is required in the replication of rate-adaptive multicast servers. It will ensure, for example, that $C1$ and $C2$ choose $S1$, while $C3$ and $C4$ choose $S2$. Without effective server selection, the benefits of replication will be lost. This leads to the problem of selecting amongst rate-adaptive multicast servers.

In this paper, we investigate this selection problem. Some fundamental issues include: 1) finding appropriate performance measures in the adaptive multicast server selection environment, and 2) studying the implications of different selection strategies on these performance measures. We first define the Static Multicast Selection Problem, in which there are static sets of clients and servers, and one needs to establish a set of multicast trees. It is possible that there is more than one tree rooted at the same server, because we allow multiple streams from each server. We explore several optimization problems based on different performance measures, with the goal to maximize utilities of clients (defined as functions of client rates) and minimize the bandwidth used by multicast trees. We prove that the general problem is NP-hard by transforming the 3-SAT problem to it and then present an interesting special case with an optimal polynomial-time solution. We design a heuristic for the general case and show that it can improve the performance over some simple strategies (e.g., Shortest,[7] Widest[8]).

We also consider the Dynamic Multicast Selection Problem, in which clients may join and leave multicast trees already established, and one needs heuristics to graft and prune the multicast trees. We design a heuristic for this dynamic case and demonstrate by simulation that it can outperform commonly used Shortest or Widest heuristics.

The rest of the paper is structured as follows. Section 2 discusses related work. Section 3 defines the Static Multicast Server Selection Problem and analyzes its complexity. A special cases is presented in Section 4. Solutions to the general static problem are given in Section 5. We define the Dynamic Selection Problem and give a solution in Section 6. Section 7 concludes the paper.

2. RELATED WORK

Server replication and selection has been extensively studied in the unicast environment.[9–11] The use of multiple multicast groups has been proposed to deal with the problem of heterogeneous receiver capacities. The idea of destination set splitting was proposed to put receivers with different capacities to different groups.[12] The sender would send to different groups at different rates controlled by the feedback from the receivers in each group. Bhattacharyya et al designed a scheme to minimize the transfer time by using multiple multicast groups in bulk data transfer.[13] Multiple multicast channels were also used in a reliable multicast scheme[14] to reduce end-host processing costs and network bandwidth usage. All of the above work focused on a single sender with multiple groups or channels. Strictly speaking, multicast server is not replicated and distributed over the network in these cases.

We undertake a study of multicast server replication and selection issues in a previous work.[6] The main focus in that work was on minimizing the cost for establishing multicast trees, assuming constant performance regardless of the tree configurations. The work gave solutions for non-rate-adaptive servers based on MDP optimization. In contrast, we focus here on the client performance of rate-adaptive servers in this paper and our solutions are aimed at optimizing the utilities of clients defined as functions of client rates.

Also related is the work on the inter-receiver fairness,[15] which shows how a single-rate multicast session is "treating" its receivers. It will be used as one of our performance measures when we consider rate-adaptive multicast server selection alternatives.

3. STATIC MULTICAST SERVER SELECTION PROBLEM

3.1. Problem Description

We use a conventional model where a network is represented by a graph $G = (V, E)$, where V denotes the set of nodes (representing hosts and routers) and E denotes the set of edges (representing logical connectivity). Edge $e \in E$ is of the form (v_i, v_j), where $v_i, v_j \in V$.

We consider a single service delivered by a set of n servers $S = \{s_1, \cdots, s_n\}$ to a static set of m clients $C = \{c_1, \cdots, c_m\}$. Assume that server s_i can send out q_i streams simultaneously. So there may exist q_i trees rooted at server s_i. An allocation of clients to servers is a function $\phi : C \to S \times Q$, where Q is the set of integers and $\phi(c_j) = \langle s_i, k\rangle$ means that client c_j is allocated to the k-th stream of server s_i. We denote the set of clients allocated to the k-th stream of server s_i as $\Omega_{ik} = \{c_j : \phi(c_j) = \langle s_i, k\rangle\}$. We aim at establishing $\sum_{i=1}^{n} q_i$ multicast trees $T_{ik} = (V_{ik}, E_{ik}), (V_{ik} \subseteq V, E_{ik} \subseteq E, 1 \leq i \leq n, 1 \leq k \leq q_i)$, such that $s_i \in V_{ik}$ and $\forall c_j \in C\ (\phi(c_j) = \langle s_i, k\rangle \Rightarrow c_j \in V_{ik})$. Note that any node (including clients) may exist in multiple trees. However, for each c_j, there exists a unique $\langle s_i, k\rangle$ such that $\phi(c_j) = \langle s_i, k\rangle$.

Let R^+ denote the set of positive real numbers. Assume a bandwidth function $B : E \to R^+$ defined on E with $B(e_i) = b_i$. The bandwidth function can be considered as the residual bandwidth currently available on the link for the service delivered by the set of replicated servers S. For adaptive multicast servers, the sending rate is determined by the minimum share of bandwidth on all links. Because a link can exist in multiple trees, a stream's share of bandwidth and, therefore, its rate depends on how we allocate the bandwidth of shared links.

Assume the rate of k-th stream of server s_i is r_{ik}. The rate of all clients selecting this stream will be r_{ik}. So the performance of these clients can be expressed as $f(r_{ik}, \Omega_{ik})$, where Ω_{ik} is the set of clients selecting k-th stream of server s_i and f is some function to be determined. The average performance of all the clients can be expressed as $\mathcal{P} = \frac{1}{m} \sum_{i=1}^{n} \sum_{k=1}^{q_i} f(r_{ik}, \Omega_{ik})$, where m is the number of clients.

We are interested in finding some collective performance measures as our optimization goals and studying how they can be expressed as a function of client rates (e.g., a special form of $\mathcal{P}$ with f defined). Our aim is to compare how different optimizing goals affects server selection algorithms and try to find algorithms that can be used for different optimizing goals with minor modifications. One simple performance measure is the average receiving rate of all clients. If $f(r_{ik}, \Omega_{ik}) = r_{ik}|\Omega_{ik}|$, then $\mathcal{P}$ is the average rate of the clients.

Another interesting measure is the collective "satisfaction" of the clients. By satisfaction we mean how a client achieves compared with its maximal possibility. For example, if a client can receive at 1 Mbps and it actually receives at 0.95 Mbps, we may say that the client is pretty much satisfied. It is not the case for a client capable of receiving at 100 Mbps. We want to use a collective measure of client satisfaction as another performance goal to optimize. The inter-receiver fairness defined in[15] measures exactly what we want here. Intuitively, we first calculate the satisfaction level of each client as the ratio of the actual rate (y_j) over maximal possible rate (r_j^*, also called *isolated rate*). Then we average the satisfaction level across all the clients. The maximal possible rate of a client is the rate at which the client can receive if there are no other clients. A simplified version of *inter-receiver fairness* (or *fairness* for short) is defined as $\frac{1}{m}\sum_{j=1}^{m} y_j/r_j^*$. $\mathcal{P}$ is the inter-receiver fairness if we define $f(r_{ik}, \Omega_{ik}) = \sum_{c_j \in \Omega_{ik}} r_{ik}/r_j^*$.

In a tree, each link carries the same bandwidth as the server rate. So the total bandwidth used by a tree is the server rate multiplied by the number of links in the tree. For tree $T_{ik} = (V_{ik}, E_{ik})$, the bandwidth used is $r_{ik}|E_{ik}|$. The average bandwidth used by one client is $\mathcal{B} = \frac{1}{m}\sum_{i=1}^{n}\sum_{k=1}^{q_i} r_{ik}|E_{ik}|$.

We want to maximize some function of $\mathcal{P}$ and $\mathcal{B}$, representing a tradeoff between performance and cost. One interesting function is $\alpha\mathcal{P} - \beta\mathcal{B}$, where α and β convert performance and bandwidth measures into common benefit and cost units. Another possibility is to maximize $\frac{\mathcal{P}}{\mathcal{B}}$, the performance per unit bandwidth. Any optimization is subject to the constraint that the total bandwidth used at each link does not exceed the link capacity. That is, for all links e_j:

$$\sum_{e_j \in E_{ik}} r_{ik} \leq b_j \tag{1}$$

In some applications, it is also possible to impose some constraints on the minimal acceptable rate of clients and servers. For example, we may require that $r_{ik} \geq \delta$ in some cases.

Other definitions of $f(.)$ are possible and they may result in some other interesting performance measures for $\mathcal{P}$. In this paper, we focus on the two measures described above, the average rate of clients and the inter-receiver fairness of clients. Before we go to the details for solving the problem, we first take a look at its complexity.

3.2. Complexity of the Problem

We are interested in knowing whether there is an efficient solution to optimize a function of $\mathcal{P}$ and $\mathcal{B}$. The problem of optimizing $\alpha\mathcal{P} - \beta\mathcal{B}$ is NP-hard by the following argument. Consider a graph in which the bandwidth of all the edges in the graph is 1.0 and assume there is only one server and it sends out only one stream. Then the value of $\alpha\mathcal{P}$ is some fixed value because the rate of the server is 1.0 and so are the rates of all the clients. With a given $\beta > 0$, the goal is reduced to minimizing $\mathcal{B}$. This is a Steiner tree problem with all edges of length 1.0, which is NP-hard.[16] Since this special case of the general problem is NP-hard, the original problem is also NP-hard.

Even if we do not consider the goal of minimizing the bandwidth used, i.e., let $\beta = 0$, the problem of maximizing $\mathcal{P}$ is still NP-hard when $\mathcal{P}$ is the average rate of clients. The 3-SAT problem[16] can be transformed to this problem in a directed graph.

Theorem. Given a network $G = (V, E)$, a bandwidth function B defined on E, n multicast servers $s_1, \cdots, s_n \in V$, m clients $c_1, \cdots, c_m \in V$, the following problem is NP-hard: Is there an allocation of clients to streams of servers with the k-th stream of server s_i operating at rate r_{ik} such that the total rate of all the clients is greater or equal to R, subject to the link capacity constraint in equation (1)?

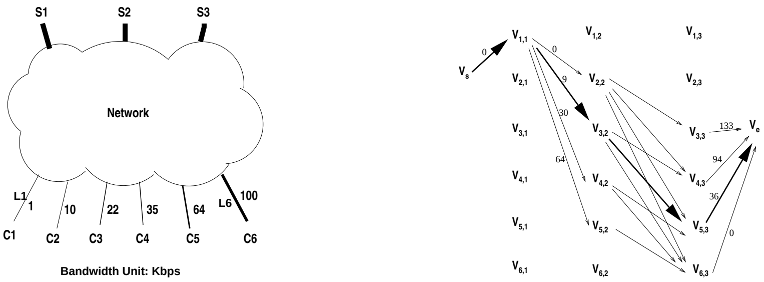
Proof. See Appendix A.

3.3. Approaches to Solving the Static Problem

Since the general static multicast server selection problem is NP-hard, we are unlikely to be able to find an efficient algorithm that gives an optimal solution. Before we explore heuristic solutions to the general problem, we are interested in finding a simplified version of the problem that may arise in practical situations and can be solved efficiently.

We mentioned that the problem consists of two parts. One is *selection*, which chooses a server for each client. Another is the *rate allocation* for servers. If two or more multicast trees share a link, the rate of each stream is determined by how we split the bandwidth of the shared link.

In the special case, we consider a situation where the bottleneck for each client is the unshared access link directly connected to it. The rate of this link can be assumed to be the maximal possible rate for each client. We assume that the other network links have enough bandwidth to support an arbitrary allocation, therefore, we do not need to



(a) The problem (b) The solution

Figure 2. An example

consider the rate allocation of shared links. After each client chooses a stream, the rate is set to be the lowest rate of all the clients selecting it. We give an optimal solution to this selection problem in the next section.

We will consider the problem of *rate allocation* on shared links in subsection 5.1. We give an optimal algorithm which will be used by the selection heuristics discussed in Section 5.

4. A SPECIAL CASE

Currently end users connect to the Internet via a variety of access technology. While the bulk of Internet clients today access the network over modems with different speeds, others have a faster access link (e.g., ISDN, ADSL). Compared with access links, the backbone network usually has a higher bandwidth. In this special case we consider these kinds of scenarios, in which each client's access link presents the bandwidth bottleneck while other network links have abundant bandwidth. For example, in Figure 2(a), client $C1$ has a bottleneck at access link $L1$ with 1 Kbps and client $C6$ has a bottleneck at access link $L6$ with 100 Kbps. The selection problem reduces to determining a grouping of clients and an allocation of a stream to each group.

In Figure 3(a), we compare the average rate of each algorithm. Assume client c_j $(1 \leq j \leq m)$ can receive at rate r_j^*, which is the isolated rate of client j, as defined in Section 3. We assume that there are n streams possible. These n streams can be sent by one server, or by n servers with each server sending one stream, or other combinations in between. The rate of each stream is the minimum rate of all the clients selecting it. If we order the clients such that $r_j^* \leq r_{j+1}^*$ $(1 \leq j \leq m-1)$, we can formulate the problem as follows: Given m clients with rates r_j^* $(1 \leq j \leq m)$ in increasing order, select n rates from them as stream rates. Assume that the n indices marking the selected rates are k_i $(1 \leq i \leq n)$. We let $k_1 = 1$ because we assume that each stream must adapt to the lowest rate of clients selecting it and every client must select one of the stream. In other words, we have a constraint $(r_i \geq r_1^*)$ on the minimum rate of clients. If we define $k_{n+1} = m+1$, we can let client $k_i \leq j < k_{i+1}$ select stream i and its effective rate is $r_{k_i}^*$. Thus the selection problem reduces to picking the n stream rates (indices) amongst the m client rates.

We have identified two goals:

1) Maximize the average rate of clients: $\frac{1}{m}\sum_{j=1}^{m} y_j$. For a given m, this problem reduces to maximizing $\sum_{i=1}^{n} r_{k_i}^*(k_{i+1} - k_i)$.

2) Maximize the fairness: $\frac{1}{m}\sum_{j=1}^{m} y_j/r_j^*$, where the fairness is defined as the average of the actual rate over the isolated rate. For a given m, this problem reduces to maximizing $\sum_{i=1}^{n}\sum_{j=k_i}^{j=k_{i+1}-1} r_j^*/r_{k_i}^*$.

4.1. Optimal Allocation Algorithms

The brute force solution to the problem involves trying every possible way to select $n-1$ stream rates from $m-1$ client rates. When $n << m$, the complexity is exponential $O(m^n)$.

We develop an algorithm with complexity $O(nm^2)$, by first transforming to a shortest path problem and then using Dijkstra's algorithm to solve it. We present an algorithm for solving the maximal rate problem, and also show how to modify it for the maximal fairness problem.

The ideal total rate for m clients is $\sum_{j=1}^{m} r_j^*$. Instead of maximizing the actual total rate of clients directly, our algorithm solves the problem by minimizing the deviation from the above ideal rate.

We represent each possible choice of stream rates from amongst client rates using a graph, with the vertices arranged in rows and columns as follows. There is one column of vertices for each stream; there is one row of vertices for each client. Vertex $v_{j,i}$ in row j and column i represents the potential for stream i to select the rate of client j. Figure 2(b) illustrates the graph for the example in Figure 2(a).

The edges in the graph are used to represent the performance associated with the choice of particular rates for two adjacent streams. That is, an edge from $v_{j,i}$ to $v_{k,i+1}$ $(j \leq k)$ represents that stream i selects the rate of client j while stream $i+1$ selects the rate of client k. The weight on this edge is the "deviation" from ideal for clients $j, \cdots, k-1$ that results from this choice. These clients will all be assigned to stream i with rate r_j^*; all except client j will therefore receive at a rate lower than their isolated rate.

A solution is represented by a path through this graph that includes one vertex per column, thus selecting one client rate per stream. With the edges and weights as described, a shortest path corresponds to an optimal solution.

Formally, we define the *deficit* from i to j $(i \leq j)$ as the difference between the isolated rates of clients i and j, i.e., $D(i,j) = r_j^* - r_i^*$. If client j selects a stream sending at rate r_i^* (note $r_i^* \leq r_j^*$), the wasted capacity by j is $D(i,j)$. The *cumulative deficit* from i to j (denoted as $D_C(i,j)$) is the sum of the deficits from i to all k $(i \leq k \leq j)$, or $D_C(i,j) = \sum_{k=i}^{j} D(i,j)$. Define $D_C(i,j) = 0$ if $i \geq j$.

We define a deficit digraph $G_d = (V_d, E_d)$, where $V_d = \{v_{j,i}\} \cup \{v_s, v_e\}$ $(j = 1, \cdots, m, i = 1, \cdots, n)$ and E_d is given as follows. Connect v_s to $v_{1,1}$ with an edge of length 0. For $1 \leq j \leq k \leq m, 1 \leq i \leq n-1$, connect $v_{j,i}$ to $v_{k,i+1}$ with an edge of length $D_C(j, k-1)$. For $1 \leq j \leq m$, connect $v_{j,n}$ to v_e with an edge of length $D_C(j,m)$. For the rates of clients in Figure 2(a), we can generate a graph in Figure 2(b) (not all edges are shown in the figure).

Using Dijkstra's algorithm, we can compute a shortest path from v_s to v_e. The path must be in the form $(v_s, v_{1,1}, v_{k_2,2}, \cdots, v_{k_j,j}, \cdots, v_{k_n,n}, v_e)$. These k_i's $(2 \leq i \leq n)$ are clients we want to partition the client set. The shortest path in the problem in Figure 2(b) is $(v_s, v_{1,1}, v_{3,2}, v_{5,3}, v_e)$ (shown as bold lines) and the optimal solution to the problem is to set streams 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.

For the maximum fairness allocation algorithm, we only need to modify the deficit from i to j to be $D(i,j) = 1 - r_i^*/r_j^*$.

4.2. Performance Results

To illustrate the use of the algorithm described above, consider a system with 400 receivers. Their isolated rates are randomly set between 0.1 to 0.9. The number of streams varies from 1 to 15. We compare five algorithms:

1. Maximize Average Rate, as described above.

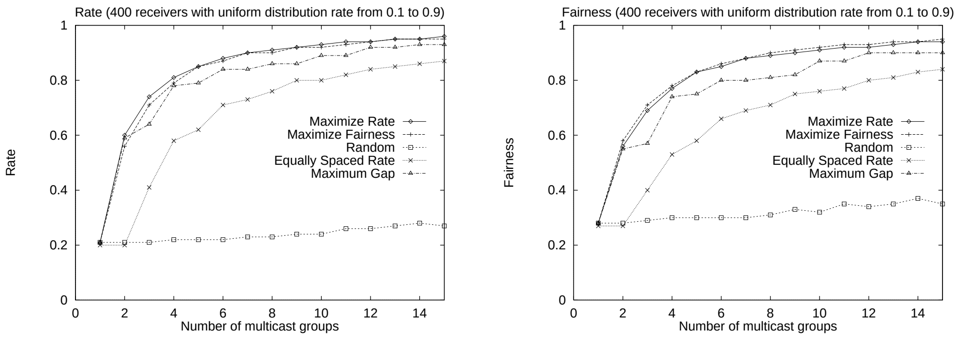
2. Maximize Fairness, as described above.

3. Random. Each client randomly selects a stream, and the rate of stream is set to the lowest of the rates of the clients selecting this stream.

4. Equally Spaced Server Rates. Assume that the range of rates is pre-determined. Each stream is assigned a rate evenly spaced from each other. Each client selects the fastest one from the set of streams that have a lower (or equal) rate than this client.

5. Maximum Gap. This is an approximation to the optimization algorithm above. It operates as follows. Assume client rates are in ascending order. Calculate the difference between the rates of two clients in adjacent positions in that order. Divide the clients into groups at the place where the difference is biggest. If there are several locations with the same biggest distance, divide arbitrarily if there are no consecutive pairs that have the same biggest distance. If there are consecutive pairs with the biggest distance, they constitute a chain. We can divide at the middle of longest chain. For the problem in Figure 2(a), the Maximum Gap solution is to set streams s_1, s_2, s_3 at rates 1, 64, 100, respectively, and allocate clients c_1, c_2, c_3, c_4 to s_1, client c_5 to s_2 and client c_6 to s_3.

In Figure 3(a), we compare the average rate of each algorithm. We use the average of isolated rates of all clients as a comparison baseline. The rate shown in the plot is the ratio of the average of actual rates of clients over the average of isolated rates. As the number of streams increases from 1 to 15, the average rate in all cases increases. The random selection algorithm does not increase much while all other algorithms demonstrate a sharp increase when the number of streams increases from 1 to 6. The maximum rate algorithm has the highest average rate, and the



(a) The average rate of clients (b) The inter-receiver fairness

Figure 3. Rate and Fairness in the Special Case

maximum fairness algorithm is very close to it. They both perform better than equally spaced rate and maximum gap algorithms.

Figure 3(b) shows the fairness of each algorithm. As expected, the maximum fairness algorithm has the largest fairness value, and the fairness of the maximum rate algorithm is very close to it. They both have a larger fairness value than maximum gap and equally spaced rate algorithms. The random selection has a very low fairness value. This is because all streams are operating at rates of some low-end clients, and all the clients are receiving at a very low rate. Therefore, both the average rate and fairness are very low when using random selection.

These results demonstrate that increased sophistication in the selection algorithm has a considerable performance payoff. The presence of rate-adaptive servers requires coordination across clients to avoid the "lowest common denominator effect" illustrated by the random result. The equally spaced rate improves upon the random result, however, the three most sophisticated schemes offer additional improvement varying from 10% to 40%.

5. HEURISTICS FOR SOLVING THE GENERAL STATIC PROBLEM

The general static problem is NP-hard, and there is no efficient optimal solution as in the previous special case. We study heuristic solutions to the problem in this section. Two obvious heuristics are:

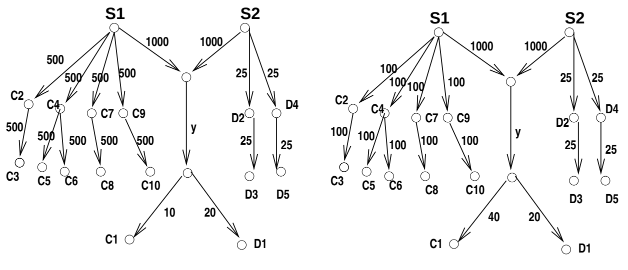
1. Shortest. Each client selects a nearest server based on the hop count. To avoid the situation that a node may be in multiple trees, the clients join sequentially. Specifically, connect a client to a multicast tree by starting from the client and following a shortest path towards the nearest server until a node already in a multicast tree is met. If no such node exists, the entire path from server to client will be added to the tree. Otherwise, a partial path will be added. There are no shared links between different multicast trees established this way. We set the server rate to be the minimum bandwidth of all the links in the server's tree.

2. Widest. First we let each client select a server that has the highest bandwidth from the server to the client. If multiple paths have the same bandwidth, we choose the shortest one. Then we set the server rates so that they are feasible and maximize the goal function using the algorithms described in the following subsection.

5.1. The Problem of Server Rate Allocation

Recall that we still have the problem of server rate allocation even if each client has selected a server according to some strategy, and the path from the server to each client is also determined. This prior allocation of clients to servers could be based on closeness to server criteria, heuristics (e.g., Widest), or perhaps some policy or pricing considerations. The problem is how to allocate the rate of each server (or stream) to maximize the average rate or the fairness index, subject to the feasibility constraints. One obvious solution is multicast max-min allocation under the definition of Tzeng and Siu.[17] However, as shall be demonstrated shortly, this does not necessarily generate the best allocation according to our measures.

We formulate the optimization problem as a Linear Programming (LP) problem and give an algorithm based on the Simplex method,[18] exploiting special properties of the problem. We use the maximal rate as an example in the problem formulation and then discuss how to modify it to solve other optimization problems.



(a) First Example

(b) Second Example

Figure 4. Experimental Settings for Rate Allocation Problem

Assume that the rate of stream s_i is r_i. Note that several streams may be sent out from the same server. For the case where we desire to maximize the average rate, the problem can be formulated as: Maximize $Z = k_1r_1 + k_2r_2 + \cdots + k_nr_n$, subject to

$$\sum_{e_j \in E_i} r_i \leq b_j, \text{for all edges } e_j \qquad \text{and} \qquad r_i \geq \delta \geq 0$$

where k_i is the number of clients selecting stream s_i and δ is some minimum rate requirement. Note that because we are given routing and client allocation to streams, the problem may have no feasible solution if too many trees share a single link, unless δ is 0. This linear programming problem can be solved by the Simplex method. We are interested in finding a simple algorithmic solution based on the special properties of the LP problem.

First we transform the formulation slightly. Let $x_i = r_i - \delta$. The original problem is transformed into a new LP problem. Maximize $Z = k_1x_1 + k_2x_2 + \cdots + k_nx_n + (k_1 + k_2 + \cdots + k_n)\delta$, subject to

$$\sum_{e_j \in E_i} x_i \leq b_j - \ell_j\delta \qquad \text{and} \qquad x_i \geq 0$$

where ℓ_j is the number of streams whose trees include the edge e_j. Obviously, if any of $b_j - \ell_j\delta$ is less than 0, then the LP problem has no solution and the original allocation problem will have no solution.

Following the Simplex method, we get the greedy algorithm. At each step, this algorithm assigns a rate to the stream with the largest number of clients that has not yet had its rate assigned. The chosen rate is as large as possible, subject to the link constraints. That stream and clients are then "removed" from the problem, and the link rates are adjusted accordingly. Assume S is the set of all streams. Here is the *greedy* algorithm:

Step 1. For each edge e_j, let $b'_j = b_j - \ell_j\delta$, where ℓ_j is the number of streams currently using this edge. If any b'_j is less than 0, then there is no solution to the original problem and exit; otherwise,

Step 2. Select a stream s_i from S with the largest k_i (number of clients). For all the edges e_j that s_i uses, select the smallest b'_j. Set x_i to this value.

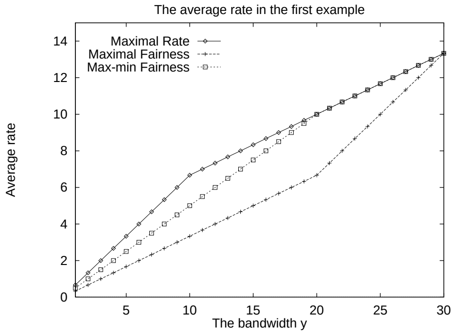
Step 3. For all the edges e_j that s_i uses, let $b'_j = b'_j - x_i$. Remove s_i from S. If S is not empty, go to Step 2.

Step 4. For stream i, set its rate to be $x_i + \delta$.

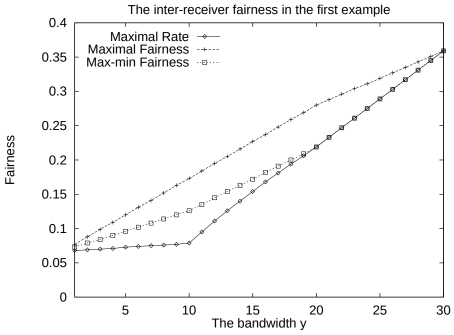
Based on the above process, we can get an optimal solution to the original problem with a guaranteed minimal rate δ for each stream. This is optimal because each step is a direct mapping from the Simplex method.

This method can be used as long as the goal function can be formulated as a linear combination of r_i. For example, we can maximize the fairness and the only modification to the above algorithm is to substitute every k_i with $\sum_{c_j \text{ chooses } s_i} 1/r_j^*$.

We show different rate allocation algorithms by two examples as illustrated in Figure 4. We have two servers $S1$ and $S2$, each sending one stream out. There are 10 clients ($C1$ to $C10$) selecting server $S1$ and 5 clients ($D1$ to $D5$) selecting $S2$. There is a shared link with bandwidth y. We change the value of y to see the effects of different algorithms. The link bandwidths are shown in the figure.

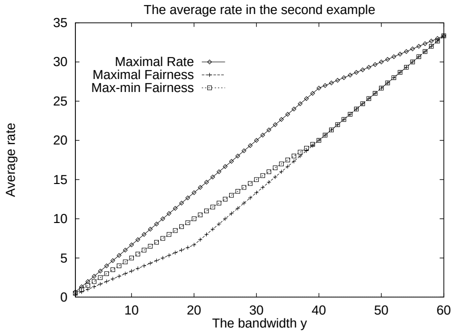


(a) The average rate of clients

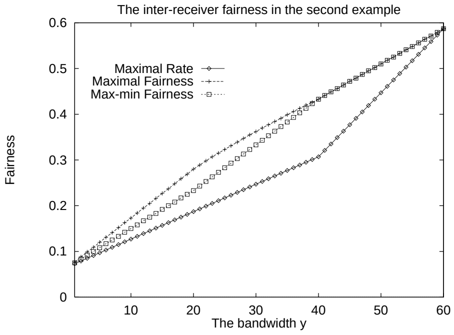


(b) The inter-receiver fairness

Figure 5. Rate and Fairness in the First Example



(a) The average rate of clients



(b) The inter-receiver fairness

Figure 6. Rate and Fairness in the Second Example

We compare three bandwidth allocation algorithms. In addition to maximal rate allocation and maximal fairness allocation algorithms, we use a multicast max-min allocation to determine the rates of servers. We measure the average rate of clients and the fairness.

In Figure 5(a) we show the average rate in the first example. The maximal fairness algorithm favors clients with a smaller isolated rate; the average rate it can achieve is smallest among three algorithms. The max-min allocation falls in the middle and converges to the maximal rate allocation when the bandwidth y is more than 20. In Figure 5(b) we show the fairness for the three algorithms. The maximal rate algorithm is worst with regard to the fairness measure. Corresponding to average rate plot, the fairness of the max-min algorithm is the same as that of maximal rate algorithm when the bandwidth y is greater than 20.

The results of allocation in the second example are given in Figures 6. The general behavior is similar to the first example. In this case, we can see that max-min allocation converges to maximal fairness allocation after the bandwidth y is greater than 40 in both the average rate and the inter-receiver fairness plots. In these examples, max-min allocation offers a compromise between maximizing average rate and maximizing fairness.

5.2. Optimized Widest Heuristic

We propose another heuristic called *Optimized Widest*. Observe that with the Widest heuristic, a client selects a path that is best for itself. However, the bandwidth of paths connecting to the same server may vary a lot. When a server has clients with considerable diversity in path bandwidth, Optimized Widest forces those clients at the low end of path bandwidth to select a different server or form another stream. This allows the particular server to operate at higher bandwidth, often with minimal penalty for the clients forced to another server.

Recall that the server set is $S = \{s_1, \cdots, s_n\}$ and client set is $C = \{c_1, \cdots, c_m\}$. Let M denote the maximum bandwidth of any edge in E.

Step 1. Add an extra node s_0 and connect s_0 with each s_i $(i = 1, \cdots, n)$ with an edge of bandwidth $M + 1$.

Step 2. Use Dijkstra's algorithm to set up a tree so that s_0 has maximum rate along this tree to any node in the graph. This consists of the following steps.

2.1) Set selected set $N = \{s_0\}$ and $\bar{N} = V - N$.

2.2) Select an edge $e = (v_i, v_j)$ such that $v_i \in N$ and $v_j \in \bar{N}$ and the residual bandwidth of e is the biggest. Add this edge to the tree.

2.3) Add v_j to N and remove v_j from $\bar{N}$.

2.4) Repeat 2.2) and 2.3) until $N = V \cup \{s_0\}$.

Step 3. From subtrees rooted at each server, pick one with the largest number of clients in the subtree. Optimize this subtree as described in the following steps.

3.1) Prune those non-client leaves (recursively).

3.2) Assume the clients selecting the server are $1, \cdots, w$ and the isolated rate of client i is r_i^* $(1 \leq i \leq w)$. Without loss of generality, we assume $r_i^* \leq r_{i+1}^*$ for $1 \leq i \leq w-1$. We determine a k such that the value $(w-k+1)r_k^*$ is maximized. We let clients $k, \cdots, w$ stay with this server and form a stream, which will operate at rate r_k^*. Clients $1, \cdots, k-1$ are pruned from the subtree.

3.3) The subtree with clients $k, \cdots, w$ is pruned to remove unnecessary leaves, if any. Reduce the bandwidth of each edge in the pruned subtree by the stream rate. These edges get a new residual bandwidth.

3.4) Remove clients in the subtree $(k, \cdots, w)$ from the client set C.

3.5) If the number of streams from this server reaches the limit of maximal number of streams from one server, then remove the server from the server set S.

Step 4. If $C = \emptyset$, stop; else goto step 2.

5.3. Performance Results with One Stream from Each Server

In this subsection, we evaluate the performance of the heuristics when there is only one stream from each server. We will consider multiple streams from each server in the next subsection.

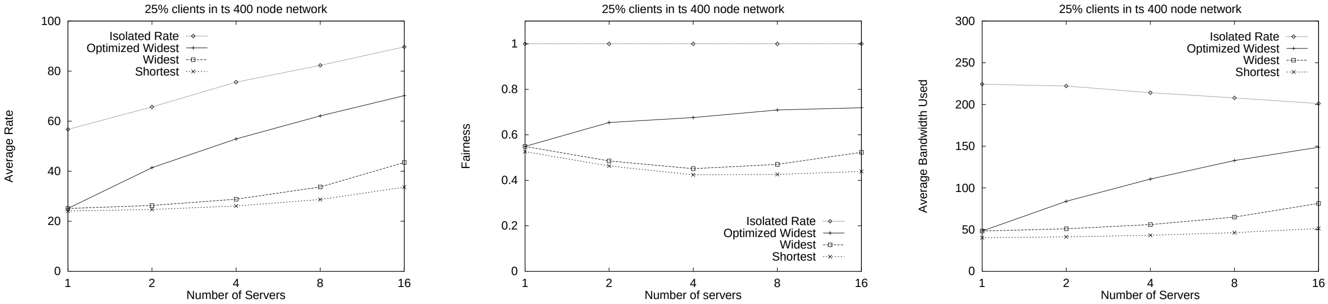
In addition to the three heuristics, Shortest, Widest, and Optimized Widest, we calculate an upper bound on the rates of clients. Each client can receive at its isolated rate if there are no other clients. Usually, this rate cannot be achieved if we have multiple clients in the session simultaneously. This is a loose bound on what we can achieve when the clients are receiving at the same time. We will compare the average rate of clients by the heuristics with the average *Isolated Rate* measure.

We generate two kinds of topologies using GT-ITM.[19] One is a random graph with 400 nodes. Another is a transit-stub graph with 400 nodes. We have 1 to 16 servers randomly placed in the graph. We experiment with two kinds of client densities: either 5% of the nodes are clients or 25% of the nodes are clients, also randomly placed in the network. The results we report are averages over a hundred simulations with randomized link bandwidths and client/server placement. This is intended to give a level of confidence in how these heuristics will perform (on average) in realistic settings. We show the results for the transit-stub graph with 25% of nodes being clients. Results in other cases are similar and can be found in.[20]

In Figure 7, we have three performance measures, Figure 7(a) shows the average rate of all clients. As the number of servers increase, the average rate increases in all cases. The Widest heuristic has a higher average rate than the Shortest heuristic, and the Optimized Widest further improves over the Widest if the number of servers is more than one. The average rate of the Optimized Widest is much closer to Isolated Rate than Widest and Shortest.

Figure 7(b) shows the fairness. From the definition in Section 3.1, it is obvious that the fairness in Isolated Rate case is always one. The Widest heuristic has a higher fairness value than the Shortest heuristic, and the Optimized Widest improves upon this when the number of servers is greater than one. Note that the fairness in the Shortest and Widest cases decreases when the number of servers increases from one to two, then to four. This is because the isolated rate increases faster than the actual rate obtained by these heuristics, thus, the average of ratio of the actual rate over the isolated rate decreases. Similar phenomenon is also observed in other cases.

Figure 7(c) shows the bandwidth used. As expected, Widest uses more bandwidth than the Shortest, and the Optimized Widest uses more bandwidth than the Widest. As the number of servers increases, the bandwidth used by



(a) The average rate of clients (b) The inter-receiver fairness (c) The bandwidth used

Figure 7. Transit Stub Topology with 25% Clients

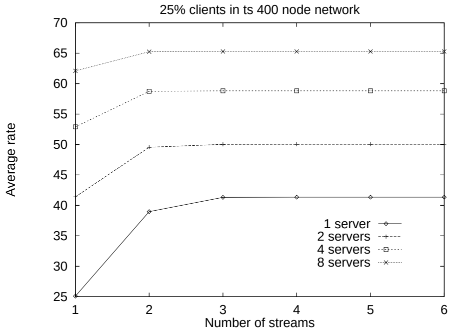


Figure 8. The case of having multiple streams

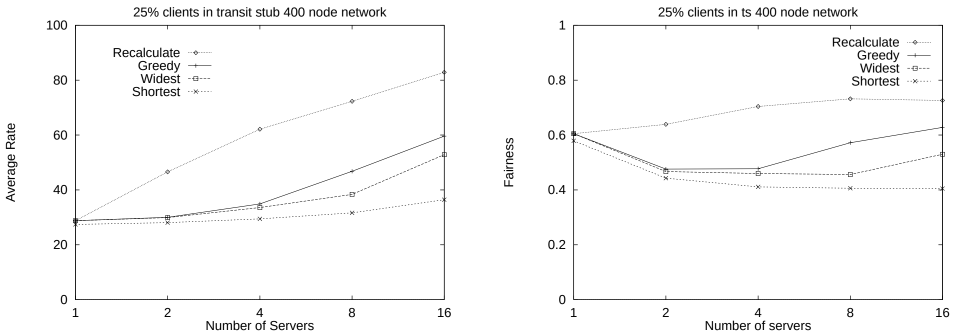
three heuristics also increases. This is because the rate of servers and clients can be higher when we have more servers. We notice that in our simulation, when the number of servers increases beyond a certain point, the bandwidth used will decrease. This is because in those situations, the average rate cannot increase much, but the distance from a server to a client becomes shorter. Therefore, the bandwidth used decreases because the number of hops from a server to its clients is smaller. The bandwidth used by Isolated Rate case decreases when the number of servers increases. This is because it may find a shorter path when there are more servers.

5.4. Performance Results with Multiple Streams from Each Server

Figure 8 shows a more interesting case, in which each server can send multiple streams. Thus it is possible that multiple trees are rooted at the same server. We use the Optimized Widest to generate the average rate of clients. We let the number of streams change from 1 to 6 and show the average rate. We experiment with 1 to 8 servers. Generally, the average rate of clients increases when the number of streams from one server increases from 1 to 4. However, when we further increase the number of the streams to higher than 4, the average rate of clients cannot be increased. This is because the bandwidth on the outgoing links from the server is limited and therefore, even if the server can support more streams, they cannot reach out to clients. While, in our example, the average rate cannot be increased by increasing the number of streams from one server beyond 4, we can have a higher average rate by having more servers. As shown in Figure 8, the average rate is increased from around 25 to more than 60 when the number of servers is increased from 1 to 8.

6. DYNAMIC PROBLEM

In many multicast applications, the set of clients changes dynamically. When a client joins or leaves a multicast session, messages are exchanged between routers to establish or tear down the appropriate paths. For the dynamic server selection problem, we assume that there are multiple multicast trees established and a client wants to join one of them. Once the client joins, the server rate will be adapted, if necessary, based on the new path. We compare several joining heuristics. When a client leaves the multicast session, we use a simple heuristic that prunes links not shared by other clients. The server rate may also change when a client leaves.



(a) The average rate of clients

(b) The inter-receiver fairness

Figure 9. Transit Stub Topology with 25% Clients

We consider four measures: the average receiving rates of clients, the fairness, the bandwidth used by multicast trees, and the join and leave cost. The calculation of performance measures in the dynamic case is a bit different from the static case. Since the trees change over time, we must include the time factor. Assume that we are interested in the time interval T, in which the system goes through N configurations of multicast trees, with each configuration lasting for t_i $(1 \leq i \leq N)$. Assume the number of clients during time t_i is n_i, the average rate is $\bar{r_i}$. We define time-normalized overall average rate as $\frac{\sum_{i=1}^{N} t_i n_i \bar{r_i}}{\sum_{i=1}^{N} t_i n_i}$. Other measures are calculated in a similar way.

6.1. Heuristics for Solving the Dynamic Problem

We have two simple heuristics that a client can use to select from a set of existing trees.

1. Shortest: Select an in-tree node that is closest to this node based on the hop count.

2. Widest: Select an in-tree node that has the highest bandwidth path to the joining node. If there are multiple paths with the same bandwidth, we choose the shortest one.

We also propose another heuristic called *Greedy*.

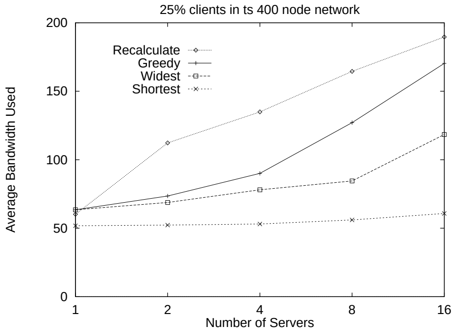
First, observe that each in-tree node v_k may be in more than one multicast tree. For each tree i, it has an associated rate $r_{k,i}$. From the node wanting to join the multicast session, we can get a widest path to each in-tree node v_k. Assuming the bandwidth of this path is b_k, we compare b_k and $r_{k,i}$. If $b_k \geq r_{k,i}$, then the path from node v_k to the joining node is not a bottleneck and the rate will be $r_{k,i}$ if the client joins this multicast group. If $b_k < r_{k,i}$, then the path from node v_k to the joining node is a bottleneck and the rate will be b_k if the client joins this multicast group. We notice in this case that the rate of the server is reduced because of this client join.

Based on the above observation, the *Greedy* heuristic will let a client join a multicast group with the biggest benefit value. We define the benefit value to be $r_{k,i}$ if $b_k \geq r_{k,i}$ and $b_k - n_i(r_{k,i} - b_k)$ if $b_k < r_{k,i}$, where n_i is the number of clients already in the multicast group for this tree. By doing so, we can make the largest increase from the existing rate.

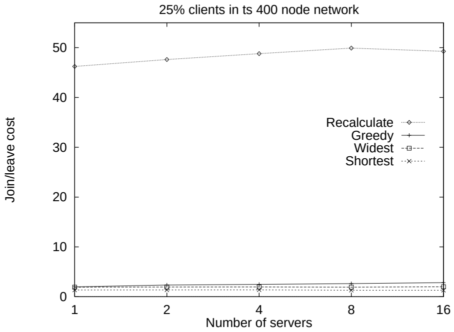
6.2. Simulation Results

For the dynamic problem, we compare the performance of three heuristics, Shortest, Widest, and Greedy. As a comparison baseline, we calculate the performance measures with the Optimized Widest heuristic described in the static case. It reallocates clients to servers from scratch whenever a client joins or leaves. This frequent tree reconfiguration has high overhead and potentially disturbs existing clients, hence it is not particularly practical.

Two kinds of topologies, random and transit-stub, are generated by using GT-ITM. There are 400 nodes in each graph, among which 1 to 16 nodes are servers. The time of a client in the multicast session is exponentially distributed and so is the time outside. By adjusting the average time in and out of the multicast session of clients, we can control the average number of clients in the multicast session to be either 5% or 25% of nodes in the graph. Again we only show the results for the transit-stub graph with 25% nodes being clients. Interested readers can find similar results in other cases in.[20]



(a) The bandwidth used



(b) The join/leave cost

Figure 10. Transit Stub Topology with 25% Clients

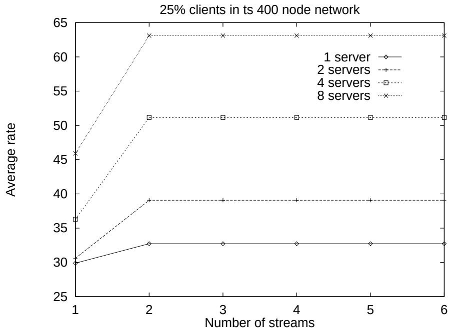


Figure 11. The case of having multiple streams

Figure 9 and Figure 10 compare the results when we allow only one stream from each server. Figure 9(a) shows the average rate of clients. We find that the Widest has a higher average rate than the Shortest heuristic and our proposed Greedy heuristic improves over Widest. The Recalculate method can adjust the tree structure when a client joins or leaves, so it can get an even higher average rate.

The fairness in Figure 9(b) and the bandwidth used in Figure 10(a) have the same relative ranking as the average rate. The fairness in the Shortest, Widest and Greedy cases decreases when the number increases from one to two, then to four. The reason is that the rate generated by the heuristics does not increase as fast as the isolated rate.

Figure 10(b) shows the the join/leave cost. We calculate the hop count of grafting and pruning trees and get an average value over time in each case. The Widest and Greedy heuristics have a higher join/leave cost than the Shortest heuristic because they may choose a longer path with a higher bandwidth. In the Recalculate case, when the number increases from one until eight, the join/leave cost increases. This is because the clients may be allocated to a farther away server. When the number increases from eight, the join/leave cost decreases because it is possible to allocate clients to a closer server, considering we have more servers. One point worth mentioning is that while the Recalculate method can get a much higher average rate and fairness, it has a very high join/leave cost, compared to other heuristics. So it may be infeasible to use the Recalculate method in the dynamic case. Among the three heuristics, our proposed Greedy heuristic has much better performance, while the cost is close to others.

Figure 11 show the results generated by the Greedy heuristic when each server can send multiple streams. We let the number of streams change from 1 to 6 and the number of servers change from 1 to 8. In each case, the average rate of clients increases when the number of streams from one server increases from 1 to 2. However, when we further increase the number of the streams to higher than 2, the average rate of clients cannot be increased. We notice in our experiments with other topologies that the average rate can be increased until the number of streams is increased up to 3. Here again we find that we can have a higher average rate by having more servers, when a server has saturated output link capacity.

7. CONCLUDING REMARKS

Multicast server rate adaptation is an approach to utilize the network resources efficiently and improve performance perceived by clients. We explore techniques of using replication to enhance the capacity of rate-adaptive multicast servers. Specifically we study the problem of how to allocate clients to replicated adaptive multicast servers in order to optimize the performance of clients. We prove that the general selection problem is NP-hard by transforming the 3-SAT problem to it. We then present and give an optimal solution to a special case. The Optimized Widest is proposed to solve the general static selection problem and its performance is demonstrated to be better than other heuristics (e.g., Shortest and Widest). We also study the dynamic selection problem for adaptive multicast servers and propose a Greedy heuristic, which is shown to be able to improve over other dynamic selection heuristics by simulation. In future work we are interested in developing practical protocols to implement selection heuristics.

REFERENCES

1. C. Diot, W. Dabbous, and J. Crowcroft, "Multipoint communication: A survey of protocols, functions and mechanisms," *IEEE JSAC* **15**, April 1997.
2. Rodriguez and E. Biersack, "Continuous multicast push of web documents over the Internet," *IEEE Network Magazine* **12**, pp. 18–31, Mar-Apr 1998.
3. A. Dan, D. Sitaram, and P. Shahabuddin, "Scheduling policies for an on-demand video server with batching," in *Proceedings of ACM Multimedia 94*, pp. 15–23, 1994.
4. S. Floyd, V. Jacobson, C.-G. Liu, S. Mccanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking* , Nov 1996.
5. X. Li and M. Ammar, "Video multicast over the Internet," *IEEE Network Magazine* , April/May 1999.
6. Z. Fei, M. H. Ammar, and E. W. Zegura, "Optimal allocation of clients to replicated multicast servers," in *Proceedings of ICNP'99*, Oct. 1999. Toronto, Canada.
7. R. Guerin, A. Orda, and D. Williams, "QoS routing mechanisms and OSPF extensions," *Internet Draft (work in progress) draft-guerin-qos-routing-ospf-00.txt* , November 1996.
8. Z. Wang and J. Crowcroft, "Quality-of-Service routing for supporting multimedia applications," *IEEE JSAC* **14**, pp. 1228–1234, 1996.
9. A. Myers, P. Dinda, and H. Zhang, "Performance characteristics of mirror servers on the Internet," in *Proceedings of INFOCOM'99*, Mar. 1999. New York.
10. S. Seshan, M. Stemm, and R. H. Katz, "SPAND: Shared passive network performance discovery," in *Proc 1st Usenix Symposium on Internet Technologies and Systems (USITS '97)*, Dec. 1997. Monterey, CA.
11. R. L. Carter and M. E. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proceedings of INFOCOM'97*, 1997. Kobe, Japan.
12. M. H. Ammar and L.-R. Wu, "Improving the throughput of point-to-multipoint arq protocols through destination set splitting," in *Proceedings of INFOCOM'92*, pp. 262–71, May 1992.
13. S. Bhattacharyya, J. Kurose, D. Towsley, and R. Nagarajan, "Efficient rate-controlled bulk data transfer using multiple multicast groups," in *Proceedings of IEEE Infocom'98*, March 1998. San Francisco, CA.
14. S. Kasera, G. Hjalmtysson, D. Towsley, and J. Kurose, "Scalable reliable multicast using multiple multicast channels," *IEEE/ACM Transactions on Networking* **8**, June 2000.
15. T. Jiang, M. H. Ammar, and E. W. Zegura, "Inter-receiver fairness: A novel performance measure for multicast ABR sessions," in *Proceedings of ACM SIGMETRICS 98/Performance 98*, June 1998. Madison, Wisconsin.
16. M. R. Garey and D. S. Johnson, eds., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, 1979.
17. H.-Y. Tzeng and K.-Y. Siu, "On max-min fair congestion control for multicast ABR service in ATM," *IEEE JSAC* **15**, April 1997.
18. W. L. Winston, ed., *Operations Research: Applications and Algorithms, 3rd ed.*, International Thomson Publishing, 1994.
19. K. Calvert, M. Doar, and E. W. Zegura, "Modeling Internet topology," *IEEE Communications Magazine* , June 1997.
20. Z. Fei, M. H. Ammar, and E. W. Zegura, "Selection of replicated adaptive multicast servers," Tech. Rep. GIT-CC-00-18, Georgia Tech, 2000.

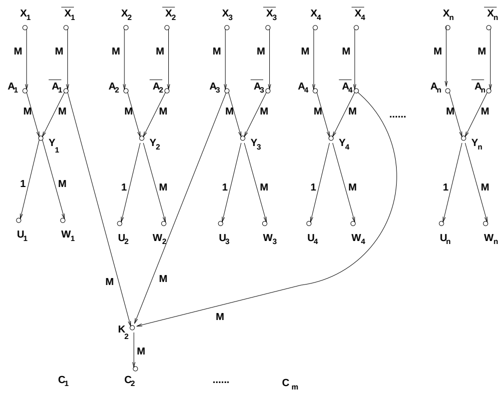


Figure 12. NP-hard proof of multicast server selection problem

APPENDIX A. NP-HARDNESS PROOF OF THE ALLOCATION PROBLEM

Theorem. Given a network $G = (V, E)$, a bandwidth function B defined on E, n multicast servers $s_1, \cdots, s_n \in V$, m clients $c_1, \cdots, c_m \in V$, the following problem is NP-hard: Is there an allocation of clients to streams of servers with k-th stream of server s_i operating at rate r_{ik} such that the total rate of all the clients is greater or equal to R, subject to the link capacity constraint in equation (1)?

Proof. The 3-SAT problem is a well-known NP-hard problem. We show that the 3-SAT problem can be transformed to the allocation problem.

Here is the 3-SAT problem: We are given a set of Boolean variables, $X_1, \cdots, X_n$. A literal is a variable X_i or its complement $\bar{X}_i$. Also given is a set of clauses C_j, where each clause contains three literals from three distinct variables. The problem is: Can each variable be set to True or False so that all clauses are true?

The transformation is as follows. Given any instance of 3-SAT problem, with n variables, and m clauses, construct a graph $G = (V, E)$ as shown in Figure 12. Set M to be some big number (like e^{m+n}). The node set V contains X_i and $\bar{X}_i$ $(i = 1, \cdots, n)$, C_j $(j = 1, \cdots, m)$ and auxiliary nodes $A_i, \bar{A}_i, Y_i, U_i, W_i$ $(i = 1, \cdots, n)$ and K_j $(j = 1, \cdots, m)$. For each variable X_i, we have an edge from X_i to A_i, an edge from A_i to Y_i, an edge from $\bar{X}_i$ to $\bar{A}_i$, and an edge from $\bar{A}_i$ to Y_i. Each has bandwidth M. Add an edge from Y_i to U_i with bandwidth 1 and an edge from Y_i to W_i with bandwidth M. For each clause C_j, add an edge from K_j to C_j with bandwidth M. If C_j contains literal X_i, then add an edge from A_i to K_j; if C_j contains literal $\bar{X}_i$, then add an edge from $\bar{A}_i$ to K_j. In the figure, we show a clause C_2 contains $\bar{X}_1$, X_3, $\bar{X}_4$. The servers are X_i and $\bar{X}_i$. The clients are U_i, W_i and C_j. The problem is to decide whether we can find an allocation with total rate $(n + m)M + n$. It is clear that the time to compute the transformation is polynomial.

Assume we can find an allocation with total rate $(n+m)M+n$. Because of constraints imposed by the bandwidth of the edge coming to them, the maximal possible rate of U_i is 1, the maximal possible rate of W_i and C_j is M. To achieve the total maximal rate of $(n + m)M + n$, they must all operate at the maximal rate. For any i, U_i and W_i can only receive from servers X_i and $\bar{X}_i$, so either X_i or $\bar{X}_i$ must have a stream with rate M and another have a stream with rate 1. These two streams cannot come from the same literal (X_i or $\bar{X}_i$) because the bandwidth from X_i to A_i and from $\bar{X}_i$ to $\bar{A}_i$ can only be M.

If X_i has a stream of rate M and $\bar{X}_i$ has a stream of rate 1, we let X_i be True. If X_i has a stream of rate 1 and $\bar{X}_i$ has a stream of rate M, we let X_i be False. Therefore, all literals having a stream of rate M are True. The fact that each C_j must have rate M implies that at least one of literal in C_j must have a stream of rate M. Therefore, every clause must be True.

Conversely, we assume that we can set variables to True or False, so that all the clauses are True. Let U_i select a server (from X_i and $\bar{X}_i$) that is False and W_i select the other one that is True. For any clause C_j, one of its literals must be True because the clause is True. Let C_j selects one of its literals that are True. By verifying against condition (1), we can see following rate allocation is feasible. If X_i is True, we set the rate of X_i to be M and the rate of $\bar{X}_i$ to 1. If X_i is False, we set the rate of X_i to 1 and the rate of $\bar{X}_i$ to M. So U_i has rate 1, W_i has rate M and C_j has rate M. So the total rate is $n + Mn + Mm = (n + m)M + n$. This completes the proof.