Optimal Allocation of Clients to Replicated Multicast Servers

Zongming Fei   Mostafa H. Ammar   Ellen W. Zegura

Networking and Telecommunications Group, College of Computing
Georgia Institute of Technology, Atlanta, GA 30332
{fei,ammar,ewz}@cc.gatech.edu

Abstract

In this paper we investigate multicast server selection problems. First we give a definition of the static multicast server selection problem*, in which we assume a set of static clients and multicast servers and consider how one might produce an optimal allocation of the clients to the servers. We use a transformation method for deriving multicast server selection algorithms from traditional multicast routing algorithms. To investigate the dynamic behavior of client join and leave and the cost incurred during the process, we next define the* dynamic multicast server selection problem*, in which the clients join and leave the multicast session dynamically. The goal is to produce an optimal allocation of clients to servers with an emphasis on how this allocation behaves over time. We formulate the problem as a Markovian Decision Process (MDP). Our analysis of the problem leads to two heuristics which we use to propose a simple selection algorithm. Our simulation compares the performance of our proposed algorithm with other multicast server selection algorithms.*

1 Introduction

Server replication is an approach that is often used to improve the scalability of a service. One of the important factors in the efficient utilization of replicated servers is the ability to direct client requests to the "best" server, according to some optimality criteria. Server replication and selection have been studied extensively in the context of unicast services [1, 2, 3], i.e., those in which a server communicates with clients in a point-to-point fashion. Typically, selecting among unicast servers is done with some goal in mind, e.g., minimizing client response time or balancing server load.

Recently, there have been several proposals for multicast services in which a server delivers information to multiple clients simultaneously. Such proposals include multicasting of web content [4, 5], multicast-based video services [6], multicasting of database content [7] and broadcast disks [8]. The goal of many of these proposals is to use multicast to enhance the ability of the service to handle a large number of clients economically.

There are two types of multicast services:

• *On-Demand Multicast Services:* These servers receive explicit requests from clients. The requests are aggregated according to some criteria to decide which data objects should be multicast.

• *Continuous Multicast Services:* These servers continuously multicast content without needing to receive explicit client requests.

Multicast servers may be replicated for several reasons:

• To distribute the load among on-demand multicast servers (much in the same way that replication is used to distribute the load among unicast servers).

• To balance the "feedback" load on the servers or on entities along the multicast tree from the servers. Examples of this feedback include acknowledgements required in many reliable multicast protocols [9] or loss information required for adaptive video transmission [10].

• To improve quality of service provided to clients and reduce the delivery cost. From a receiver'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 receiver 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.

In this paper, we study the multicast server selection problem. Our goal in this initial study is to consider some fundamental aspects of this problem. Towards that end we define two main problems:

• *Static Multicast Server Selection*: In this problem we assume a set of static clients and multicast servers and consider how one might produce an optimal allocation of the clients to the servers. For this problem, we analyze its relationship with traditional multicast routing, in which we establish one "optimal" tree from the server to all the receivers in the multicast group.

When the multicast server is replicated, we need to establish an "optimal" set of trees with one server as a root for each tree. Our approach is to formulate several graph-theoretic problems that capture various multiple-tree optimization criteria and consider solution approaches that are based on single-tree multicast routing.

• *Dynamic Multicast Server Selection*: In this problem we assume a fixed set of multicast servers but a dynamic set of clients with a given join/leave behavior. Again our focus is to produce an optimal allocation of clients to servers with an emphasis on how this allocation behaves over time. In this problem one needs to give consideration to the cost of joining and leaving a multicast group. We formulate the dynamic multicast server selection as a *Markovian Decision Process* (MDP) [11]. This allows us to factor in the stochastic join/leave behavior of receivers and to explore the tradeoff between the cost of the multicast tree(s) and the transition cost of establishing and removing links from the tree(s). Because of the limited scalability of MDP solution approaches, we are only able to use them to consider small problems. We use the properties of the optimal solutions generated from these small examples to develop a set of dynamic server selection heuristics.

The rest of this paper is structured as follows. In Section 2 we give definitions of several variations of the static multicast server selection problem and discuss solution approaches. Section 3 presents a transformation method to solve a particular static selection problem, the minimum cost allocation problem. A model of the dynamic multicast server selection problem is given in Section 4. We use the MDP method to solve the problem and analyze several issues in dynamic server selection in Section 5. Section 6 presents a dynamic multicast sever selection algorithm based on the heuristics summarized from the analysis of solutions to the examples using MDP method. The results of our simulation of various selection algorithms are presented in Section 7 and the paper is concluded in Section 8.

2 Static Multicast Server Selection Problems and Solution Approaches

We use a conventional model where a network is represented as a graph $G = (V, E)$, where V denotes the set of nodes (representing hosts or 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 assume G is a simple graph. We define a cost function on E as $C : E \rightarrow \Re^+$ and a delay function on E as $D : E \rightarrow \Re^+$, where $\Re^+$ is the set of positive real numbers. A path between node s and d is a sequence of nodes $(v_0, \cdots, v_k)$ such that $(v_i, v_{i+1}) \in E$ $(0 \leq i \leq k-1)$ and $s = v_0, d = v_k$. The set of paths between nodes s and d in graph G is denoted as $P_G(s, d)$.

Allocation Problems Assume there are n servers $S = \{s_1, \cdots, s_n\}$ and m receivers $R = \{r_1, \cdots, r_m\}$ in graph $G = (V, E)$. We aim to determine n multicast trees $T_i = (V_i, E_i), (V_i \subseteq V, E_i \subseteq E, 1 \leq i \leq n)$, such that $s_i \in V_i$ and $\forall r_j \in R\ \exists T_i = (V_i, E_i)(r_j \in V_i)$. That is, every receiver is in a multicast tree. For simplicity, we assume each receiver is in exactly one tree. We identify two major allocation problems based on their goals.

• The *minimum cost allocation problem* is to find n multicast trees such that the total cost of tree links is minimum, i.e., minimize $\sum_{i=1}^{n} \sum_{e \in E_i} C(e)$. This is a generalization of the Steiner tree problem [12]. Our approach is to establish a relationship between the minimum cost allocation problem and the minimum Steiner tree problem. We use a transformation method for deriving minimum cost allocation algorithms from corresponding algorithms for the minimum Steiner tree problem, which will be described in detail in Section 3.

• The *minimum delay allocation problem* is to find n multicast trees such that the average delay from servers to receivers is minimum. We first define the *distance* between node s and d in graph G as $dist_G(s, d) = \min_{p \in P_G(s,d)} \sum_{e \in p} D(e)$. Then the minimum delay allocation problem is to minimize $\sum_{i=1}^{n} \sum_{\substack{v_j \in V_i \\ v_j \in R}} dist_{T_i}(s_i, v_j)$. Since the total delay is minimized, the average delay is also minimized for a given m. This problem can be solved by using a shortest path tree algorithm. We can simply allocate each receiver to its nearest server.

Problem Variations If we impose further constraints on the trees established, we may formulate several other allocation problems. For example, the *balanced allocation problem* has constraints on the number of receivers allocated to each server. The *delay bounded minimum cost allocation problem* has a constraint on the longest delay from servers to receivers in the minimum cost allocation problem. They are discussed in more detail in [13].

3 A Solution to the Minimum Cost Allocation Problem

Here we focus on a solution to the minimum cost allocation problem. We can reduce the multiple tree minimum cost allocation problem to a single tree routing (Steiner tree) problem with the following transformation method. This allows us to use any of the Steiner tree algorithms [14]. Recall there are n servers $S = \{s_1, \cdots, s_n\}$ and m receivers $R = \{r_1, \cdots, r_m\}$ in graph $G = (V, E)$.

1) Add a virtual node s_0 to the graph and connect it to all servers with edges of cost 0. We get a new graph $G' = (V', E')$, where $V' = V \cup \{s_0\}$ and $E' = E \cup (\bigcup_{i=1}^{n} \{(s_0, s_i)\})$.

2) Solve the minimum cost allocation problem with a single server $\{s_0\}$ and m receivers $R = \{r_1, \cdots, r_m\}$. This is a minimum Steiner tree problem for G' with a set of compulsory nodes $\{s_0\} \cup R$.

3) Let T' denote the Steiner Tree. Remove any edge (s_0, s_i) $(1 \leq i \leq n)$ if it is in T'. We get a forest $\mathcal{T}$,

which is the final allocation of receivers to servers.

Lemma If the cost function satisfies $C(e) > 0$ for any $e \in E$ and T' is a minimum Steiner tree for G' with a set of compulsory nodes $\{s_0\} \cup R$, then the forest $\mathcal{T}$ (T' restricted to V) is a minimum allocation for G.

4 The Dynamic Multicast Server Selection Problem

In most multicast applications, the membership of receivers changes dynamically. A receiver can join or leave a multicast session at any time, which generally requires the addition (grafting) or removal (pruning) of paths from the multicast routing trees. This is normally accomplished through the use of a *multicast routing protocol* which uses the exchange of messages among multicast routers to establish or tear down the appropriate paths (e.g., [15, 16, 17]). Grafting and pruning, therefore, incurs a cost that is typically proportional to the length of the path being added or removed. This join/leave cost can be high if the receivers join and leave frequently or if the paths being grafted or pruned are long. In this dynamic environment, client allocation to multicast servers must, therefore, consider both the tree cost and the join/leave cost. The following is a model for the dynamic server selection problem that captures both the tree cost and join/leave cost.

Given a network $G = (V, E)$, we assume there are n servers $S = \{s_1, \cdots, s_n\}$ and m potential receivers $R = \{r_1, \cdots, r_m\}$. We assume that each receiver remains a member of the multicast session for a random period of time (the *time-in-session*) with distribution $B(t)$ and mean $1/\mu$. Once it leaves the multicast session it re-joins after a random period (the *time-to-join*) with distribution $A(t)$ and mean $1/\lambda$. While a receiver is in a multicast session, it is receiving from exactly one server and belongs to the multicast group to which this server is transmitting. Note the distinction between *a multicast session* and *a multicast group*.

A cost rate function $F : E \to \Re^+$ is defined on the edges of the graph G. This represents the cost *per unit time* of using this edge as a part of a multicast tree. Another cost function $H : E \to \Re^+$ is defined on the the edges of G. This represents the cost of pruning or grafting this edge onto a multicast tree.

Each receiver r_j $(1 \le j \le m)$ can select any of the servers. At any given time, the *receiver state* can be described as $J =< j_1, \cdots, j_m >$, where $0 \le j_k \le n$ for $1 \le k \le m$. If $j_k = 0$, the receiver r_k is not in the multicast session at this time. If $j_k > 0$, the receiver r_k is in the multicast tree rooted at the server s_{j_k}. A transition of state occurs when a receiver joins or leaves the session. We say that a state J *induces a set of trees* $T_i = (V_i, E_i)$, $i = 1, \cdots, n$, that are subgraphs of G. The specific set of trees will be a function of the multicast routing in use. The set $\mathcal{T} = \{T_i;\ i = 1, \ldots, n\}$ completely specifies the state of the system. In the analysis to follow we assume that shortest path tree (SPT) routing is used to form the trees from the receiver state J [1].

There are two kinds of costs incurred in the dynamic join/leave process, *state occupancy* cost and *transition* cost.

• **State Occupancy Cost.** The state occupancy cost is a measure of the cost incurred in operating the system in a particular state and is proportional to the time the process is in the state and the costs (from F) of the edges in the trees being used. More formally the state occupancy cost in state $\mathcal{T}$, which contains the trees $T_i = (E_i, V_i)$, $i = 1, \cdots, n$, is given by $\text{OCost}(\mathcal{T}, t) = t \sum_{i=1}^{n} \sum_{e \in E_i} F(e)$, where t is the time spent in state $\mathcal{T}$.

• **State Transition Cost.** The transition cost is incurred during the transition from one system state to another, as a result of a receiver joining or leaving. It is a function of the costs (derived from H) of edges removed or added as a result of the transition between the two states. Consider the transition from state $\mathcal{T}_1$ to $\mathcal{T}_2$. This results in a change from a set of induced trees $T_{i1} = (E_{i1}, V_{i1})$ to another set of induced trees $T_{i2} = (E_{i2}, V_{i2})$, $i = 1, \cdots, n$. If we define the set of edges $L_i(\mathcal{T}_1, \mathcal{T}_2) = (E_{i1} \cup E_{i2}) - (E_{i1} \cap E_{i2})$, the total cost of this transition is given by $\text{TCost}(\mathcal{T}_1, \mathcal{T}_2) = \sum_i \sum_{e \in L_i(\mathcal{T}_1, \mathcal{T}_2)} H(e)$.

Assume that the initial configuration of the tree is $\mathcal{T}_0$ and there are r transitions, with transition i changing the state from $\mathcal{T}_i$ to $\mathcal{T}_{i+1}$, where $i = 0, 1, 2, \cdots, r-1$. Assume that the duration in state $\mathcal{T}_i$ is t_i, and $t = \sum_{i=0}^{r} t_i$. The expected total cost of running the system for the t time units is given by

$$\text{TotCost}(t) = E[\alpha \sum_{i=0}^{r} \text{OCost}(\mathcal{T}_i, t_i) + (1-\alpha) \sum_{i=0}^{r-1} \text{TCost}(\mathcal{T}_i, \mathcal{T}_{i+1})] \quad (1)$$

where α is a weight that can be used to change the relative cost of occupancy and transition.

If the transition from some receiver state $J = < j_1, \cdots, j_m >$ to another receiver state $J' = < j'_1, \cdots, j'_m >$ is triggered by receiver ℓ joining the session, then we know $j_\ell = 0$ because ℓ is not a member of any group before the join, and $j'_\ell > 0$ because ℓ must have selected some server and joined the multicast tree. If the transition is triggered by receiver ℓ leaving the session, then we know $j_\ell > 0$

[1] Note that because of the possible routing ties, the combination of J and SPT routing does not unambiguously define the system state. For some network topologies, however, it is possible to use the receiver state J in conjunction with the use of SPT routing to completely define the state of the system. Note that in general routing changes can result in system state changes without any corresponding changes to the receiver state. In our analysis we do not allow such changes and all system state changes occur as a result of receiver state changes.

and $j'_\ell = 0$. For any other receiver, the membership remains the same before and after the (join or leave) event. Therefore, if $j_k = 0$ then $j'_k = 0$ and if $j_k > 0$ then $j'_k > 0$ for all $k \neq \ell$. Whether j'_k is equal to j_k or not for $j_k > 0$ $(k \neq \ell)$ distinguishes two cases. One is the model *without switching*, in which a join/leave of one receiver will not change other receivers' selection of servers, i.e., $j'_k = j_k$ for all $j_k > 0$ $(k \neq \ell)$. The other is the model *with switching*, in which a join/leave of one receiver can change other receivers' selection of servers, i.e., j'_k can be different from j_k. Note that allowing server switching can lead to a lower cost of operation. Some applications may not allow for server switching, however.

In each state, a *decision* specifies a next state for each possible join/leave event. Assume the current receiver state is $J =< j_1, \cdots, j_m >$. If $j_\ell = 0$, the receiver ℓ may join a group. A decision in this case would indicate the server to which this new receiver should be allocated. In addition if our system allows switching, a decision would include any changes to the allocation of other receivers as a result of this particular join event occurring. If $j_\ell > 0$, the receiver ℓ may leave the group. A decision for this leave event would be specified if switching is allowed and would dictate the allocation changes to be made to remaining receivers. The new state J' is, therefore, determined by the decisions made, with different decisions leading to a different sequence of states over time.

A *policy* is a set of decisions, with one decision for each possible system state. The optimal policy is the one that minimizes the expected value of the cost per unit time, i.e., minimizes $\liminf_{t\to\infty} \frac{\mathrm{TotCost}(t)}{t}$.

5 Solution to the Dynamic Multicast Server Selection Problem

If receiver time-to-join and time-in-session distributions ($A(t)$ and $B(t)$) are exponential, then we can define a Markovian Decision Process (MDP) for the above model. Howard [11] developed a policy-iteration algorithm which is guaranteed to produce a policy that minimizes the cost for an MDP. The difficulty in applying the algorithm is that the state space grows exponentially with the number of receivers in the problem.

Our approach is to use the MDP method to investigate the dynamic nature of the multicast server selection problem for small systems and to develop solution heuristics that can be applied to large systems.

5.1 Tradeoff Between State Cost and Transition Cost

We now illustrate the interesting properties of the optimal policy using the simple network setup shown in Figure 1. There are two servers, s_1 and s_2, and two receivers, r_1 and r_2. Each edge is labeled with a single value that happens to be both the cost rate and the graft/prune cost. We will rely on the weight α in equation (1) to give us any further normalization of these costs. We assume the arrival rate and departure rate are 0.1. We can set α to 1 to get the best policy when we only consider the state occupancy cost. Similarly we get the best policy for cases when only the transition cost is considered by setting α to 0. We experiment with α from 0 to 1 with step 0.1 to see how the best policy changes as the weight of the state occupancy cost increases relative to the transition cost.

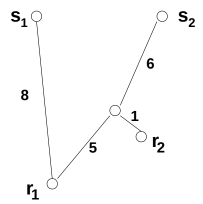

FIGURE 1: The network setup

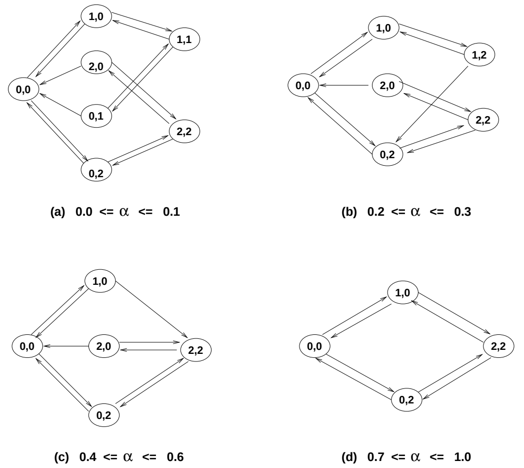

FIGURE 2: The tradeoff between state occupancy cost and transition cost

In Figure 2, we show the evolution of the best policy in the case where switching among servers is allowed. As previously defined, the state is in the form of a tuple, which gives the server selected by each receiver. For example, $< 2, 1 >$ indicates that receiver r_1 selects server s_2 and receiver r_2 selects server s_1. A value of 0 means that the receiver is not in the multicast session. In the figure, we show only those states reachable from state $(0, 0)$ by the policy. In all cases, receiver r_1 joins s_1 if neither r_1 nor r_2 is in the session, i.e., state $< 0, 0 >$ to state $< 1, 0 >$. Similarly receiver r_2 joins s_2, i.e., state $< 0, 0 >$ to state $< 0, 2 >$. The more interesting behavior occurs when the number of receivers in the session goes from 1 to 2 (due to a join) or from 2 to 1 (due to a leave). We show four cases based on different α values. When α changes from 0 to 1.0, the best policy evolves from minimizing transition cost to minimizing state cost.

5.2 A Larger Example

We now use the MDP approach to solve a selection problem with two servers and four receivers. Again our goal is to analyze the solution to the problem

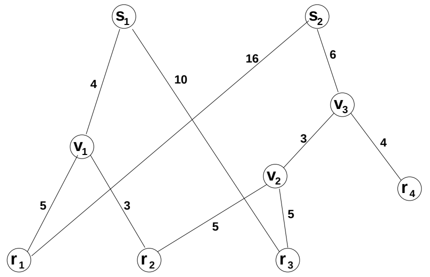

FIGURE 3: A server selection problem with 2 servers and 4 receivers

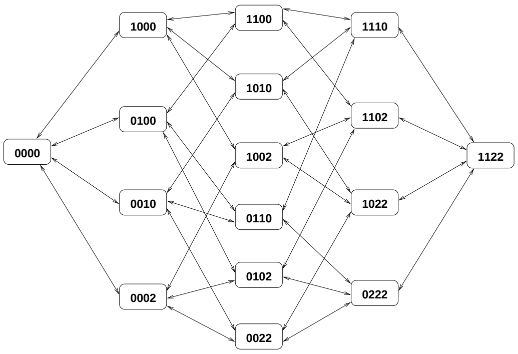

FIGURE 4: The solution to the server selection problem with 2 servers and 4 receivers

and identify heuristics that can be used to solve larger problems.

In Figure 3 we have two servers s_1, s_2, and four receivers r_1, r_2, r_3, r_4. Each edge is labeled with a single value that is both the occupancy cost rate and the graft/prune cost. We assume that the arrival rate is $\lambda = 10^{-4}$ and the departure rate is $\mu = 10^{-3}$ for all the receivers. We set the parameter $\alpha = 0.5$. In this example, we assume server switching is allowed. We use Howard's algorithm to obtain an optimal policy as shown in Figure 4, where states are in the form of tuples as previously defined.

The join and leave behavior of the receivers is as follows:

- **Join Behavior**

Receiver r_1 always selects s_1 and receiver r_4 always selects s_2. This is because selecting the other server will incur very high cost. The more interesting case is the behavior of receivers r_2 and r_3.

Consider the state transition from $< 0002 >$ to $< 0022 >$. Receiver r_3 will select s_2 if r_4 is already in the tree because the extra cost is 8 (from r_3 to v_3), which is smaller than selecting s_1 with an extra cost of 10 (from r_3 to s_1). Receiver r_3 will select s_1 if r_4 is not in the tree (e.g., the transition from $< 0000 >$ to $< 0010 >$) because the cost of selecting server s_1 is smaller than the cost of selecting server s_2. Similar is the case for receiver r_2. If r_3 is connected to server s_2, receiver r_2 will select server s_2 (e.g., the transition from $< 0022 >$ to $< 0222 >$), otherwise it will select server s_1 (e.g., the transition from $< 0000 >$ to $< 0100 >$).

An important assumption in our model is that we only consider the shortest path from a server to a receiver. So the candidate nodes that a receiver considers for server selection are those nodes in the shortest paths from all the servers to it. The above observation leads us to the following join heuristic: *A receiver will select the nearest in-tree node in its shortest paths to all servers and establish a path from that node to the receiver.*

- **Leave Behavior**

Leave is simple if we do not consider the effects of one receiver's leave on other receivers' selection. We can simply remove the path leading to this receiver that is not shared by other receivers. We now focus on how the tree changes for other receivers when one receiver leaves the tree.

Consider state $< 0022 >$, where receivers r_3 and r_4 are both connected to s_2. When receiver r_4 leaves the tree, r_3 changes from selecting server s_2 to s_1, leading to state $< 0010 >$. This is because r_3 can share links with r_4 in the original state, but the links are not shared once r_4 leaves.

Now we analyze state $< 0222 >$, where receivers r_2, r_3 and r_4 are connected to server s_2. When r_3 leaves, receiver r_2 changes from s_2 to s_1. We observe that r_2 shares a link with r_3. This indicates that when a receiver leaves, we should make an adjustment to those receivers sharing links with the departing receiver. We can reduce the cost by letting them re-select a server.

This leads us to the following leave heuristic: *When a receiver leaves, we find those receivers that share links with it and let them re-select servers.*

6 Dynamic Multicast Server Selection Algorithms

Our experience with the MDP-based solution leads us to some possible selection algorithms which we describe here. We first describe several join algorithms and then consider leave algorithms.

6.1 Join Algorithms

Nearest Server Algorithm. In the nearest server algorithm, a receiver will select the nearest server and join the multicast group associated with this server. In the worst case, this may require the grafting of the entire path from the server to the receiver. The problem with this approach is that it will select the nearest server even if there is a nearby tree. Setting up a path from a nearby tree may add less cost than setting up a path from a server to the receiver. Therefore, the cost of the tree may be higher.

Nearest Tree Algorithm. In the nearest tree algorithm, a receiver joins the multicast group whose routing tree requires the shortest path to be grafted in order to join the group. This leads to a lower cost tree, but it is hard to implement. Usually a joining receiver needs to know all the nodes in the trees in order to select the nearest one.

Nearest Node Algorithm. Based on the join heuristic described in Section 5.2, we propose this new selection algorithm. Instead of letting a receiver select the nearest server or the nearest tree, we let the receiver select the nearest node among those that are in the shortest paths from servers to the receiver and already in the multicast trees.

Let Γ be the set of nodes in the shortest paths from servers to this receiver. Let Ω be the set of nodes that are already in the multicast trees. Select the nearest node $v \in \Gamma \cap \Omega$ and join the tree by establishing a shortest link from v to this receiver.

This algorithm is simpler than the nearest tree algorithm if we do not know what nodes are already in the multicast trees, because we need consider only those nodes in the shortest paths.

6.2 Leave Algorithms

No Adjustment. When a receiver wants to leave a multicast group, we can use the standard pruning method: if the receiver is not a leaf in the tree, simply remove the receiver from the group; if the receiver is a leaf in the tree, delete all the links that are not shared by other receivers.

With Adjustment. If we allow switching, we may be able to reduce the tree cost. The idea is to let all the receivers sharing edges with the deleted node reselect a server using a join algorithm. If the receiver is not a leaf, all the receivers in the subtree rooted at this node will re-join a group (re-select a server) using previous join algorithms. If the receiver is a leaf, we search the nodes towards the root of the tree until we locate an upstream node (v) satisfying one of the three conditions.

1) It is a server. No adjustment will be made.

2) It is a node that has at least two outgoing links (including the outgoing link connecting the departing receiver). We let each receiver in the subtree rooted at v re-select a server.

3) It is a receiver in the multicast group. We let v re-select a server.

In the nearest server algorithm, the receiver's choice is independent of the state of other receivers, thus leave adjustment will not reduce the cost. We can only use leave adjustment to reduce the cost for the nearest tree algorithm and the nearest node algorithm.

7 Performance Analysis

In this section, we describe results from a set of simulation experiments in which we study the performance of our proposed algorithm with various other multicast server selection algorithms.

7.1 Experimental Settings

Network Topologies. We use the GT Internetwork Topology Models simulator [18] to generate the network topology. We experiment with 1 to 16 servers in a network with 400 nodes. By setting the time-in-session and time-to-join for receivers, we can control the average number of receivers in the multicast session to be 5% to 30% of all the nodes, or 20 to 120.

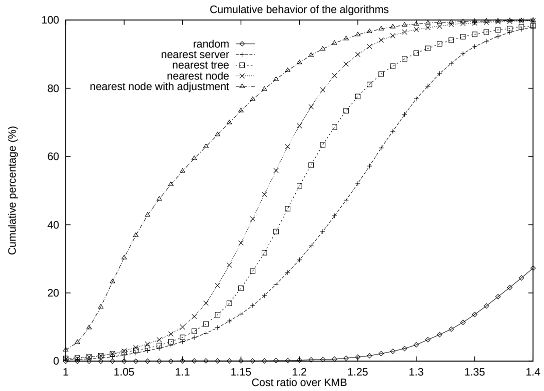

FIGURE 5: Cumulative distribution of the cost ratios

Algorithms Compared. We compare the performance of five algorithms for multicast server selection.

1) Random selection. Every receiver randomly selects a server. A tree is constructed by the shortest path tree algorithm for each server.

2) Nearest server (SPT).

3) Nearest tree without adjustment.

4) Nearest node without adjustment.

5) Nearest node with adjustment.

Metrics of Interest. Two basic metrics we use to compare the algorithms are the *average tree cost* and the *transition cost*. The network topology generated has a cost function defined on edges. The function value of an edge is used as both the cost rate and the graft/prune cost.

7.2 Numerical Results

In the first experiment we have 2 to 16 servers in the network and the percentage of receivers in the multicast session changes from 5% to 30%. When a receiver joins or leaves the multicast session, we construct a set of trees from scratch. The algorithm we use is based on the KMB algorithm [14] and generated by using the method described in Section 3 We use the cost of these trees as the baseline cost for comparing the five algorithms. We calculate the ratio of tree cost in each case by each algorithm over the tree cost generated by the KMB-based algorithm in the same settings. The cumulative distribution of this ratio for each algorithm is shown in Figure 5. The x-axis is the ratio of the tree cost and the y-axis is the corresponding percentage. We find that random selection performs worst among the five algorithms. For example, the y-value is only 27% when the x-value equals 1.4, which means, only 27% of the trees are within 1.4 times the tree cost generated by the KMB-based algorithm. All other algorithms are very close to 100% for this ratio. We can conclude that random selection for replicated multicast servers will result in very poor performance. For the other four algorithms, the nearest node algorithms increase faster than either the nearest server or the nearest tree algorithm. The nearest node algorithm with adjustment performs best. For example,

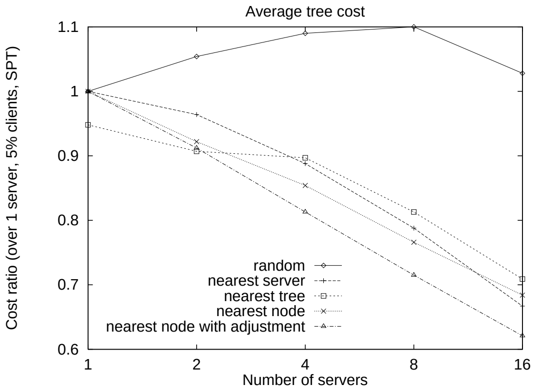

FIGURE 6: The average tree cost (5% Clients)

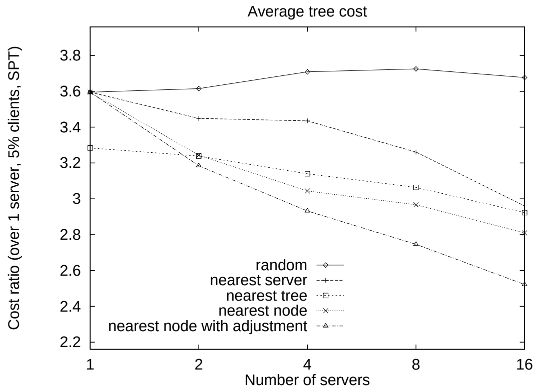

FIGURE 7: The average tree cost (30% Clients)

we can see that 78% of the trees generated by the algorithm are within 1.15 times the tree cost generated by the KMB-based algorithm, while only 20% of the trees generated by the nearest tree algorithm and less than 15% by the nearest server algorithm are within this range.

Figure 6 shows the average tree cost as a function of the number of servers. The average number of clients in the multicast session is maintained at the level of 5% of all the nodes in the network. We use the average tree cost generated by the SPT algorithm with one server as a comparison baseline and show the ratio of average tree cost over that value in each case. We can see the tree cost decreases as the number of servers increases, except in the random case. The nearest tree algorithm performs best when there is only one server. However when the number of servers increases, its average cost is larger than the nearest node algorithms. When the number of servers increases to four or more, it is worse than the nearest server algorithm. The random case generates the worst trees when the number of servers is more than one. This is because the algorithm does not consider sharing between edges and may select a longer path that is not shared with any other receiver. Note that the 16 server case is better than 8 server case because the effect of increasing the number of servers outweighs the effect of random selection, but it is still worse than the one server case. The tree costs resulting from the use of the nearest node algorithms decrease faster than other algorithms. They outperform the nearest server algorithms when there is more than one server and are better than the nearest tree algorithm when there are four servers or more. The nearest server algorithm may add extra cost because of favoring the nearest server over a nearby in-tree node. The nearest tree algorithm may lead to an unusually long path in the tree after several joins/leaves. The nearest node algorithms are a compromise between these two extremes and result in better performance in the multiple server cases.

Figure 7 shows the case when the average number of clients in the session is maintained at the level of 30% of all the nodes in the network. To make this comparable with the 5% client case, we use the ratio over the cost generated by the SPT algorithm in the one server case with 5% clients. By comparing the two plots, we can see that the average tree cost does not increase linearly with the average number of clients in the session, because of sharing of cost among clients. We also compare how the average tree cost changes with the number of servers. When there are 30% clients, the average tree cost does not decrease as quickly as in the 5% client case. This is because in the 30% client case, there are already a lot of sharing of edges between clients even with 1 server. Increasing the number of servers does not improve the performance as much as in the 5% client case. Another difference is that the nearest tree algorithm performs better than the nearest server algorithm. This is because we have a better chance of sharing for the nearest tree algorithm, and the effect of unusually long paths is reduced due to the larger number of receivers.

Figure 8 shows the join and leave cost when the average session size is 5% of nodes. We get the average cost over time and use the ratio over the cost induced by the SPT algorithm in the one server case. Generally the join and leave cost decreases as the number of servers increases, except in the random case. For the random algorithm, adding more servers does not help to decrease the join/leave cost. The join/leave costs of the five algorithms are rather similar except for the nearest node algorithm with adjustment. This is because the algorithm not only adds/removes a path for the join/leave client, it may change the path for other clients in the tree as well.

Figure 9 shows the case where the average session size is 30% of nodes. Again we use the ratio over the cost induced by the SPT algorithm in the 1 server case with 5% clients. The difference between costs with different number of servers is smaller in this case. This is because there is a higher chance that nearby nodes are already in the tree, and the effect of the number of servers is lessened. This also leads to a smaller average join/leave cost than in the 5% client case. The only exception is the algorithm with adjustment, which has

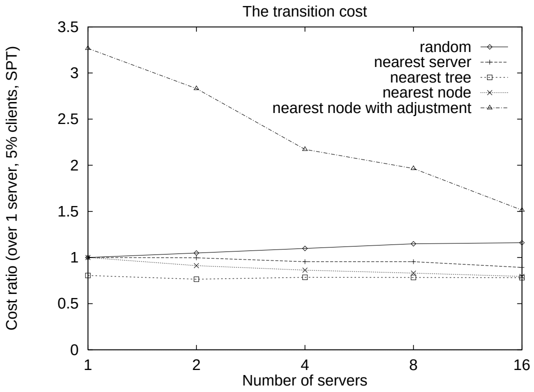

FIGURE 8: The join/leave cost (5% Clients)

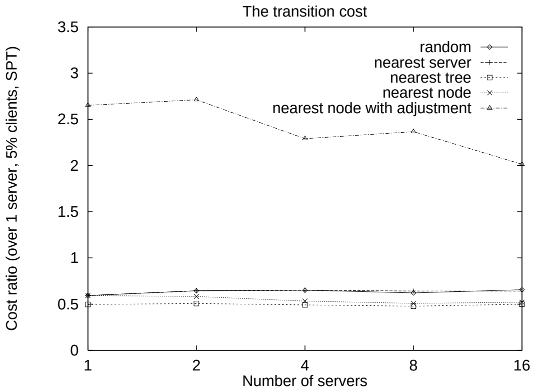

FIGURE 9: The join/leave cost (30% Clients)

higher join/leave cost for four or more servers than in the 5% client case. This can be explained by the fact that a larger number of clients in the session results in adjusting more clients in each join and leave.

8 Concluding Remarks

In this paper, we define two categories of multicast server selection problems: static selection and dynamic selection. For the static multicast server selection problem, we present a transformation method for deriving algorithms from the traditional multicast routing algorithms. For the dynamic multicast server selection problem, we formulate it as a Markovian Decision Process. We explore the tradeoff between the cost of the multicast trees and the transition cost of establishing and removing links from the trees. We analyze the properties of the optimal solutions generated from these small examples to develop a set of server selection heuristics. Our simulation shows that selection algorithms based on these heuristics can perform better than other simple algorithms, such as nearest tree and nearest server algorithms.

This has been a mostly theoretical study focusing on some fundamental aspects of the multicast server selection problem. In future work we will consider more practical aspects of this problem with the objective of developing a framework and a set of protocols that can be used by clients for server selection. We will also consider the issue of server placement and how that affects the performance of the protocols used for server selection.

References

[1] S. Bhattacharjee, M. H. Ammar, E. W. Zegura, V. Shah, and Z. Fei, "Application layer anycasting," in *Proceedings of INFOCOM'97*, 1997.

[2] R. L. Carter and M. E. Crovella, "Server selection using dynamic path characterization in wide-area networks," in *Proceedings of INFOCOM'97*, 1997.

[3] Z. Fei, S. Bhattacharjee, E. Zegura, and M. Ammar, "A novel server selection technique for improving the response time of a replicated service," in *Proceedings of INFOCOM'98*, 1998. San Francisco, California.

[4] M. Ammar, K. Almeroth, R. Clark, and Z. Fei, "Multicast delivery of web pages or how to make web servers pushy," in *Proc. of the Workshop on Internet Server Performance*, June 1998. Madison, WI.

[5] Rodriguez and E. Biersack, "Continuous multicast push of web documents over the Internet," *IEEE Network Magazine*, vol. 12, pp. 18–31, Mar-Apr 1998.

[6] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Proc. of ACM Sigcomm'97*.

[7] S. Mahajan, M. Donahoo, S. Navathe, M. Ammar, and S. Malik, "Grouping techniques for update propagation in intermittently-connected databases," in *Proceedings of the 1998 IEEE Conference on Data Engineering*, Feb. 1998. Orlando, Florida.

[8] R. Alonso, M. Franklin, and S. Zdonik, "Broadcast disks: Data management for asymmetric communication environments," in *Proceedings of the ACM SIGMOD*, May 1995. San Jose, CA.

[9] 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.

[10] X. Li and M. Ammar, "Video multicast over the Internet," *IEEE Network Magazine*, April/May 1999.

[11] R. Howard, ed., *Dynamic Programming and Markov Processes.* MIT Press, Cambridge, 1960.

[12] E. N. Gilbert and H. O. Pollak, "Steiner minimal trees," *SIAM Journal of Applied Mathematics*, vol. 16, no. 1, pp. 1–29, 1968.

[13] Z. Fei, M. H. Ammar, and E. W. Zegura, "Optimal allocation of clients to replicated multicast servers," Tech. Rep. GIT-CC-99-13, Georgia Tech, 1999.

[14] L. Kou, G. Markowsky, and L. Berman, "A fast algorithm for Steiner trees," *Acta Informatica*, vol. 15, pp. 141–145, 1981.

[15] S. Deering and D. Cheriton, "Multicast routing in datagram internetworks and extended LANs," *ACM Transactions on Computer Systems*, vol. 8, pp. 85–110, May 1990.

[16] B. M. Waxman, "Routing of multipoint connections," *IEEE Journal of Selected Areas in Communications*, vol. 6, pp. 1617–22, December 1988.

[17] M. Doar and I. Leslie, "How bad is naive multicast routing?," in *Proceedings of INFOCOM'93*, 1993.

[18] K. Calvert, M. Doar, and E. W. Zegura, "Modeling Internet topology," *IEEE Comm. Magazine*, June 1997.