Protocols for Selection Among Replicated Multicast Servers

Lenitra M. Clay
Mostafa H. Ammar
Ellen W. Zegura

Networking and Telecommunications Group
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
{lenitra,ammar,ewz}@cc.gatech.edu
(404)894-3292, Fax: (404)385-0332

Feb 21, 2002

Abstract

Transmitting data via multicast can typically generate less network traffic than if the same packet were to be unicast to each of the receivers. The network load generated by multicast applications can be further reduced by multicast server replication. For the replication to be utilized efficiently, the clients need to be directed to the "best" server according to some optimality criteria. With replicated multicast servers, the shape of the multicast tree also needs to be considered during the server selection process to reduce the join/leave overhead incurred by clients and the cost of the multicast trees. In this paper we describe and evaluate three novel multicast server selection protocols. The first is a resolver-based protocol where the client queries a resolver and the resolver returns the IP address of the "best" server. The second protocol uses the anycast service to allow clients to find the nearest on-tree multicast router. Finally, the third protocol requires router support so the clients are informed of what their join cost would be for each of the multicast servers, allowing the client to select the server with the lowest cost. We evaluate and compare these three protocols in terms of the cost of the multicast trees generated.

1 Introduction

IP multicast allows a sender to send a single packet to a group of receivers. A multicast address defines a multicast group; receivers can dynamically join and leave this group using the IGMP (Internet Group Management Protocol) [1, 2, 3]. Packets are forwarded to the multicast group according to a multicast routing protocol, e.g. PIM, DVMRP, CBT [4, 5, 6, 7] which typically constructs a multicast tree over which the packet is duplicated when the path to the receivers diverges. The copying of the packet within the network reduces the network load when compared to unicasting the packet to each receiver. This efficiency of multicast has been exploited to propose many scalable multicast services such as multicasting web content [8, 9, 10], multicast video-on-demand [11, 12, 13, 14] and multicasting database content [15]. Multicast server *replication* can further reduce the network load generated by multicast applications. Replicating multicast servers will allow for a few small multicast trees to be generated instead of one potentially large tree. Thus replication can be used to improve the scalability of the multicast service. To efficiently utilize these replicated multicast servers, however, the multicast clients need to be directed to the "best" server according to some optimality criteria.

Our target application for these multicast server selection protocols is a set of replicated multicast servers that are transmitting the same long-lived multimedia stream such as radio or T.V. broadcasts. These streams represent long-lived content distribution session. Examples of such content distribution sessions are musical concerts, sporting events and theatrical presentations.

In [16] a multicast server selection algorithm is proposed and evaluated but no selection protocols are described. In this work we propose and evaluate protocols for implementing multicast server selection. We consider three novel multicast server selection protocols that can be used to assign clients to replicated multicast servers. The protocols differ in the method by which the assignment takes place. The three protocols are:

1. A *resolver-based* approach where resolvers only need to maintain information about multicast servers and clients contact these resolvers to be told which multicast server they should contact.

2. An *anycast-based* [17] approach where the clients send their request to join the multicast group to an anycast address associated with that multicast group allowing the network based anycast routing to direct the client to the best multicast tree.

3. A *router-supported* approach where the multicast tree routers store information that can be used by clients to help determine the best multicast tree onto which to graft.

With each approach there is a tradeoff between the amount of support from the network and the performance of the selection algorithm.

Due to the long-lived nature of the streams, clients may join and leave over time making leave adjustment a desirable feature of the selection protocol. We, therefore, also consider how *leave adjustment* can be performed by each of the protocols. Leave adjustment will allow clients to re-select a server when the multicast tree changes due to another client leaving that multicast group. Since clients can dynamically join and leave the multicast group, the shape of the multicast tree changes and it may also be useful to perform leave adjustment.

The multicast server selection protocols require a cost metric to be associated with each link. This cost metric could be performance based to represent measures such as QoS or the capacity of the links, or distance based to represent a hop count distance measure. We evaluate the three multicast server selection protocols in terms of the cost of the multicast server trees generated. We find that the resolver-based protocol performs well and the protocol can be deployed in the Internet today. The anycast-based and router-supported protocols also generate low cost trees while requiring more network support. The anycast-based protocol is a promising multicast server selection protocol to use once a global anycast service is provided in the Internet. The rest of this paper is structured as follows. The next section discusses related work. In Section 3 we describe the resolver-based multicast server selection protocol. Section 4 discusses the anycast-based approach and Section 5 discusses the router-supported protocol. In Section 6 we evaluate the three approaches. Finally, we give concluding remarks in Section 7.

2 Related Work

In recent years, a great deal of research has considered the question of how to best select a server from a set of replicated unicast servers. In [18] the authors explore the use of dynamic server selection to retrieve WWW documents. In the SPAND [19] work, a system is developed to allow for passive collection of network performance information by clients within a region. In this system, clients send out performance reports summarizing the performance received by distant hosts. These performance reports are sent to a performance server which stores them in a database. Then clients wishing to determine the performance of a distant host queries the performance server. An empirical study of client-side server selection algorithms was done in [20]. Their work builds upon related studies, such as [18, 19], by examining the effects of time-of-day, client resources, and server proximity. In [21] the authors examine the use of application-layer anycasting to improve client performance when accessing replicated web servers. In their architecture an anycast group is defined to be a set of unicast or multicast IP addresses. Their design uses anycast resolvers to maintain selection metrics. Web server response time was chosen as the metric of interest, and metric collection is performed by using a server push and resolver probe technique. The client queries a resolver specifying an anycast domain name that identifies the group and the selection criteria to be used, and the resolver responds with the IP address of the server to contact.

In the unicast server selection work, since the servers communicate with clients in a point-to-point fashion, the typical goal is to assign the client to the server that minimizes response time or balances the server load. With replicated multicast servers our goal is to assign the client to the multicast server that keeps the multicast tree cost low and minimizes the join/leave cost as well. When clients join and leave a multicast group the multicast routing protocol must graft and prune branches of the multicast tree. Thus joining and leaving the multicast group incurs a cost. So client allocation to multicast servers must consider the dynamic nature of client behavior and take into account the cost of the multicast tree as well as the join/leave cost. In [16] the authors investigate algorithms to optimally allocate clients to replicated multicast servers, where the goal is to have the servers handle a large number of clients economically. The **nearest node** join algorithm performed the best out of the join algorithms evaluated. In their work a node refers to a router. The **nearest node** is the router that is on the shortest path from a server to the client, a member of the multicast group and the closest to the client in terms of the cost metric. The nearest node algorithm works by first finding all the nodes in the shortest paths from the servers to the client. Then the client selects from these nodes the one which is a part of the multicast trees and closest to the client. The authors also considered leave algorithms to allow for adjustment of the selected server.

3 Resolver-Based Multicast Server Selection

In this section we describe our resolver-based multicast server selection protocol. The resolver-based multicast server selection architecture consists of *resolvers* that maintain information about servers and assign clients to particular multicast servers. The resolvers are located close to the exit point of their domains. These resolvers obtain information about the multicast servers by listening to a multicast control group, to which all of the replicated multicast servers

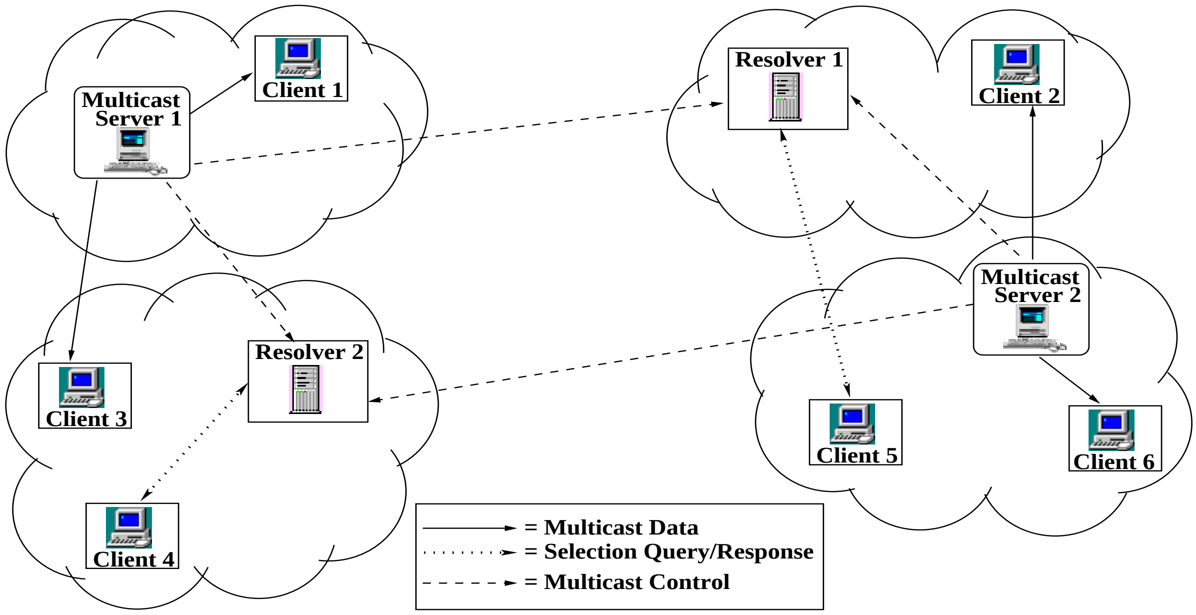


Figure 1: Resolver-Based Multicast Server Selection Architecture

periodically send. Clients send their request for the desired multicast service to the resolver responsible for their domain. This resolver will return the multicast group address and server IP address to the client along with any other necessary join information. Once the response is received, the client can join the multicast group using normal IGMP messages. Figure 1 illustrates the resolver-based multicast server selection architecture. In the figure there are four domains, two of which contain multicast servers and the other two contain the resolvers. Multicast data is sent from the servers to the multicast group members. Selection queries and responses are sent between the selecting clients and the resolver responsible for their domain. The multicast control group data flows from the multicast servers to the resolvers.

3.1 Clients and Servers

With this protocol, clients desiring to select a server:

1. Send a selection request to the resolver responsible for their domain.
2. Wait for the response from the resolver.
3. Join the multicast group of the server specified using normal IGMP messages.

The selection request specifies the type of multicast service that is being requested. For example, the particular video they wish to view. We assume that the multicast servers in a server group are providing the same service and the exact nature of the service is not a concern of this paper.

The replicated multicast servers need to:

1. Send out their multicast data as usual.
2. Send periodic information messages about themselves on a multicast control group to which the resolvers listen.

The information the servers send to the resolvers will have two parts. The first is information to be used in the selection process, such as the current video schedule, and the current sending rate of the server. The second part is membership information such as the multicast group address, the IP address of the multicast server, and a decryption key. These packets are used by the resolvers to determine a server-to-resolver cost metric. The TTL value used in these control messages should be at least as high as the TTL used for the multicast data. That way if a resolver does not hear from a server it means it should not assign clients to that server because the clients would probably not be able to receive the server's data either.

3.2 Resolvers

The resolvers in this system perform two important functions:

1. Collection of information to be used when selecting servers.
2. Processing of client requests to select a server.

In order to assign the clients to multicast servers, the resolvers need to maintain information about the servers. Our approach is to have the resolver maintain cost metric information about the servers to help in assigning clients. There are several ways that a resolver can obtain cost metric information. For example, to obtain distance information the resolver can possibly tap into an existing Internet Distance Mapping Service such as that provided by the techniques in [22], or the resolver can use the multicast traceroute tool, `mtrace` [23], to determine the physical hop count, packet rate and loss statistics for the path from the resolver to the server.

When a client request is received, the resolver:

1. Narrows the selection based upon the requested service.
2. Selects the server, from the remaining servers, that has the lowest server-to-resolver cost.
3. Returns the IP address of the server, along with any other necessary join information, to the client.

Locating the resolvers near the exit point for the domain will mean that the closest server, in terms of the cost metric, to the resolver is also the closest server to the clients contacting that resolver. Since clients contact the resolver responsible for their domain, this server selection has the effect that all clients within a domain are assigned to the same multicast server.

In order for clients to perform leave adjustment with this protocol the client would need to re-contact the resolver for the IP address of the multicast server. This is because the client does not know the shape of the multicast tree and thus can not determine when other clients have left the multicast group. The resolver will respond to this query as usual and return the IP address of the server it believes the client should contact at this point in time. If the server address is different from the address the client was previously assigned, the client decides whether or not to switch to the new server. How often the client should re-query the resolver is not clear. Allowing for leave adjustment with this selection protocol would lead to increased protocol overhead from clients re-contacting the resolvers.

4 Anycast-Based Multicast Server Selection

In this section we describe our anycast-based multicast server selection protocol. RFC 1546 [17] describes an Internet anycasting service for IP and proposes its use for service discovery. Our anycast-based multicast server selection protocol uses anycast to allow the client to join the server tree that has the nearest on-tree router in terms of the cost metric. All servers register with their designated routers (DRs) telling the designated routers to join the specified anycast group. The designated routers are the routers responsible for sending IGMP membership queries for the network. The anycast address is advertised along with the multicast group address associated with the replicated servers. When a client needs to select amongst the servers, it sends a join request to the anycast group. This join request will be forwarded to the nearest anycast group member.

As the join request travels to the nearest anycast group member, transient join state is setup in the routers along the path. When the join request is received by the nearest anycast group member it sends a join acknowledgement back towards the client. This join acknowledgement contains the multicast source address and will traverse the reverse path of the corresponding join request. The reception of the join acknowledgement by routers maintaining transient join state causes them to:

1. Add the multicast source address reported in the acknowledgement to the state being maintained.

2. Start an anycast timer whose expiration will cause them to join the anycast group.

3. Forward the packet downstream to the client using the stored join state.

Anycast is used by the clients to find the nearest on-tree router to graft onto and setup branches of the multicast tree. Routers forward the multicast packets over the on-tree interfaces for the multicast source specified in the multicast packet. Keepalive messages sent between the routers are used to maintain the multicast tree. When a router finds that it has no more downstream members it leaves the anycast group, removes the join state, and stops sending keepalive messages. The DRs of the multicast servers remain members of the anycast group as long as the servers continue to send multicast data.

4.1 Leave Adjustment

In order to perform leave adjustment, routers that find they are now sending data on a single outgoing interface will send an adjust message down this interface as well as upstream towards the multicast source. The adjust message notifies receivers downstream of this prune point that the multicast tree has changed as a result of a receiver leaving the group. These receivers may need to select a different multicast server. Routers receiving an adjust message from their parent and those initiating the sending of the adjust message will:

1. Stop announcing membership in the anycast group.

2. Start a leave adjust timer to remove the join state if a new join acknowledgement is not received.

3. Forward the adjust message downstream over the on-tree interfaces for the specified multicast source.

Thus the adjust message will travel to all receivers downstream of the prune point. An adjust message received by a downstream receiver causes the receiver to send out a re-join request which travels to the nearest anycast group member.

The adjust message also notifies upstream routers that adjustment of the multicast tree is taking place. Routers that have only one child interface and receive an adjust message will:

1. Start a leave wait timer.

2. Stop announcing membership in the anycast group.

3. Forward the adjust message upstream towards the multicast source.

Expiration of the leave wait timer will cause the router to re-announce membership in the anycast group. When the adjust message reaches a router that has multiple child interfaces the adjust message is silently discarded and that router will continue announcing its anycast membership. If the adjust message reaches the designated router of the multicast server it is discarded and the designated router continues to announce its anycast membership.

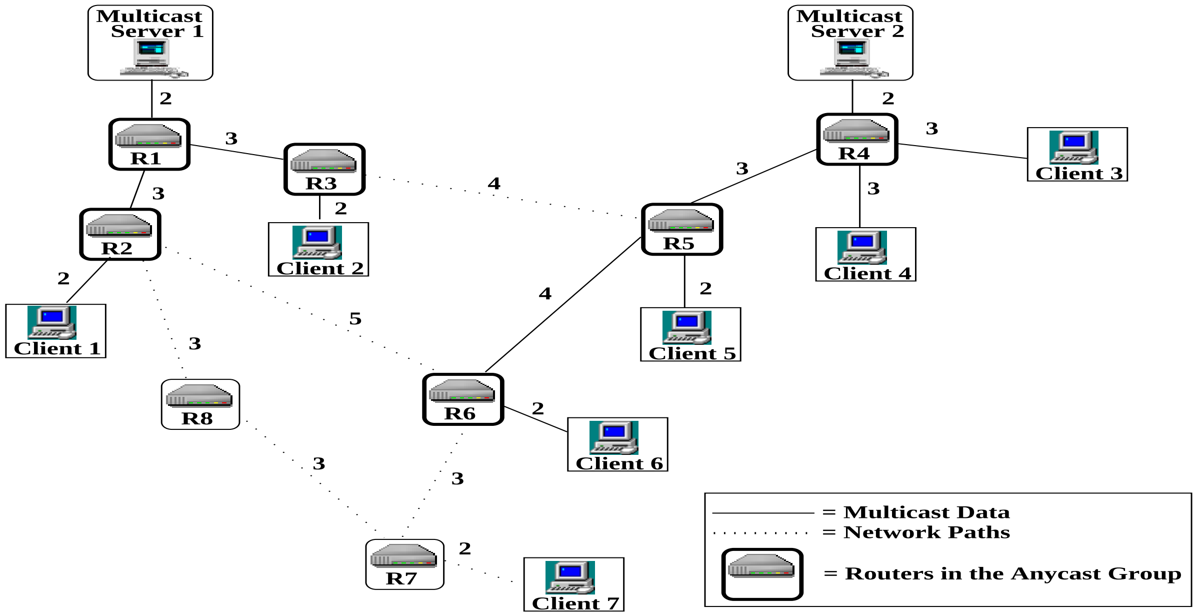


Figure 2: Anycast-Based Server Selection Leave Adjustment

An example will illustrate the need for the adjust message to travel both up and down the multicast tree. Figure 2 shows two multicast servers with six multicast clients receiving the multicast data. Routers announcing anycast group membership are indicated in bold. The solid lines indicate the current multicast trees. Each edge is labeled with the cost metric value. Routers $R1$ and $R4$ are the DRs for $servers1$ and 2 respectively. $Client7$ wants to select amongst the two servers and sends an anycast join request. The request is forwarded to the nearest anycast group member which is $R6$. $R6$ sends an acknowledgement back to $Client7$ which causes the routers on the path from $R6$ to $Client7$ to start an anycast timer. When the anycast timer expires these routers will all join the anycast group. Thus $Client7$ has joined the multicast tree of $Server2$. Later, $Client6$ leaves the multicast group. $R6$ will notice that it is now forwarding over a single interface and send an adjust message down that interface. $R6$ will also send an adjust message upstream. When the adjust message reaches $R5$ it discards the message and continues announcing anycast group membership. The adjust message sent downstream from $R6$ will reach $Client7$ and $Client7$ will send a re-join message upstream. Now since routers on the path from $R6$ to $Client7$ are no longer announcing anycast membership, the re-join request will be sent to $R2$ which is the closest anycast group member to $Client7$. $R2$ generates an acknowledgement and sends it back down to $Client7$. Reception of the acknowledgement by routers on the path from $R2$ to $Client7$ will cause them to join the anycast group. Thus $Client7$ has switched to the tree of $Server1$.

During the adjustment period, the downstream receivers are still capable of receiving the multicast data because the join state has not yet been removed. The adjust message must travel both up and down the multicast tree to cause the routers along the path to stop announcing anycast membership so that a new multicast server may be selected. Receivers located in subtrees rooted upstream of the prune point will not select a new multicast server. A problem created by the leave adjustment is that join requests issued during an adjustment period may not travel to the nearest on-tree router since some routers temporarily stop announcing anycast membership. For example, after $Client6$ leaves the multicast group, $R6$ and $R7$ continue to forward the multicast data to $Client7$ during the adjustment (until the expiration of the leave adjust timer) but they are not announcing anycast group membership. Future work will look into how to handle new join requests during the adjustment.

4.2 Example Use with GIA

Our anycast-based multicast server selection protocol requires the use of global anycast groups to allow the servers to exist anywhere in the Internet. In [24] a scalable global IP-anycast (GIA) architecture is described. With GIA, border routers (BRs) in edge domains cache routes to groups that are frequently accessed by the domain's users. Edge domains classify anycast groups according to their popularity and membership, and route to them accordingly. There are three anycast group classifications:

1. internal - the domain has at least one internal member. Unicast intradomain routing is used to route to the group.
2. popular - the domain has users that frequently access the group. A cached route is used to forward to this group.
3. unpopular - the domain has no local members and the group is not frequently accessed. Packets destined to this group are forwarded toward the home domain, the Internet domain that owns the anycast address.

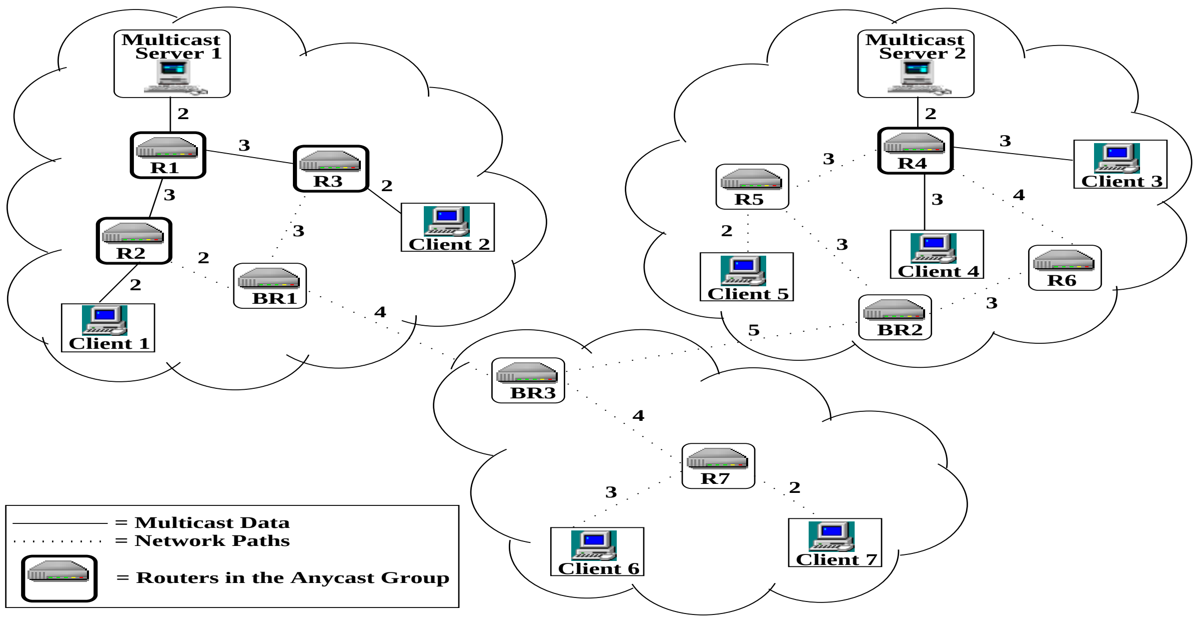


Figure 3: Anycast-Based Multicast Server Selection Example

An example will illustrate the workings of this multicast server selection protocol with GIA. Figure 3 shows two multicast servers that each serve two multicast clients. The multicast trees are indicated by the solid lines. Routers $R1$ - $R4$, indicated in bold, are also members of the anycast group associated with the multicast service. Edges are labeled with the cost metric. Routers $R1$ and $R4$ are the DRs of the two servers. $Client5$ wishes to chose a server and sends a join request to the anycast group address. This join request will be sent to $R4$ since it is the closest anycast group member. So $Client5$ joins the tree of $Server2$. $R4$ will send a join acknowledgement downstream which will cause $R5$ to start an anycast timer to begin announcing membership in the anycast group. Later, $Client6$ decides to join the multicast group. Since the domain does not have an anycast group member the anycast message will reach $BR3$. This border router will route the anycast message using GIA either by a learned route or towards the home domain of the anycast group. Since this is the first anycast message from this domain, and thus the group is an unpopular group, the message is sent towards the home domain. The domain of multicast $Server1$ has been chosen as the home domain and so $Client6$ joins the tree of $Server1$. $BR3$ sends the join request to $BR1$ which sends it to router $R2$. The join acknowledgement will follow the reverse path back to $Client6$ and router $R7$ will start announcing anycast membership. $BR3$ will then mark the domain as having an internal anycast group member. Now when $Client7$ wants to join the multicast group the join request is sent to $R7$ so that $Client7$ also joins the tree of multicast $Server1$. Some time later when $Client6$ leaves the multicast group, $R7$ will realize it is forwarding over a single link and send an adjust message downstream to $Client7$. $R7$ will also send an adjust message upstream towards the multicast sender. Now $BR3$ will no longer have an internal anycast group member and when the new join request is received from $Client7$ it routes towards the home domain again. Thus $Client7$ will remain a member of the multicast group served by $Server1$.

4.3 Reducing the Anycast Membership

The cost of anycast group membership can be further reduced by allowing only the branch point routers and the designated routers to join the anycast group. The protocol described above would need to be modified as follows:

- Routers receiving a join acknowledgement do not join the anycast group. Reception of an acknowledgement causes the routers maintaining transient join state to add the multicast source address reported in the acknowledgement to the state being maintained.
- As described above, the join request would be sent to the anycast group and travel towards the nearest anycast group member. However, when a router with join state for the multicast receives a join request on a different interface from which it is currently forwarding, that router responds with the join acknowledgement and joins the anycast group. This router has now become a branch point router and will forward the multicast packets over more than one interface. Only routers with join state that are not waiting for timers to expire can respond to join or re-join requests.
- Routing updates would need to be modified so that the anycast routers advertise their

cost to the multicast server as well. This is needed to ensure that join requests are sent to the anycast router with the lower cost to a multicast server. In the protocol described above this was not needed because all multicast routers were also members of the anycast group. With only branching point routers advertising anycast group membership, there could be cases where a client's first hop router is equal cost from two anycast routers but one has a lower cost to the multicast server and is therefore preferred to the other router. In such a situation the cost to the multicast server can be used to decide which anycast router to store in the routing table.

Now, with fewer routers as members of the anycast group, leave adjustment is easier to perform. Routers that find they are now sending data on a single outgoing interface will send an adjust message downstream. With the reduced anycast group membership there is no need to send the adjust message upstream. The next upstream anycast member is a branch point router who is sending over more than one interface and does not need to stop announcing anycast membership. If a new server is selected, routers on the path between the prune point and the next upstream branch point will eventually stop receiving keepalives and remove their join state. The adjust message is handled by downstream routers and receivers as described in section 4.1.

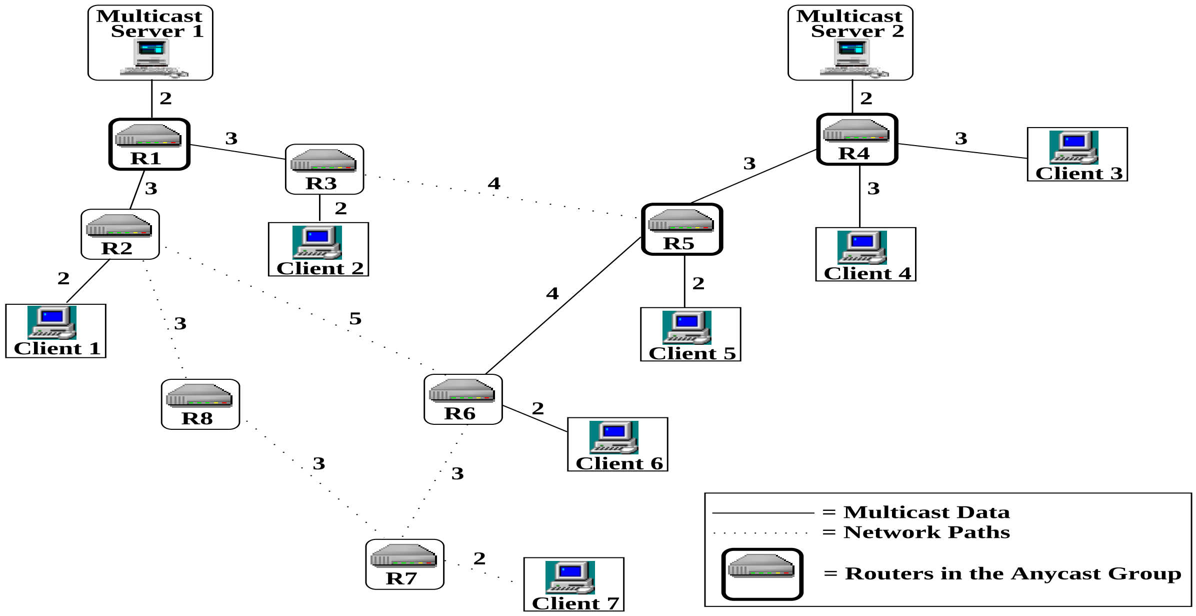


Figure 4: Reduced Anycast Membership Example

We have modified the network from Figure 2 to show the reduced anycast membership. Figure 4 shows three routers announcing anycast group membership instead of the six routers shown in Figure 2. These three routers represent the branch point routers. $Client7$ wants to select amongst the two servers and sends an anycast join request. The request is forwarded to the nearest anycast group member which is $R5$ in this case. However this is not the closest router that is a part of the multicast tree, that is router $R6$. When the join request reaches $R6$ it realizes that it is already forwarding the data on a separate interface and generates an acknowledgement. $R6$ will also start an anycast timer whose expiration will cause $R6$ to join the anycast group. The join acknowledgement travels back to $Client7$ causing the routers on the path from $R6$ to $Client7$ to update the join state with the selected server address. Thus $Client7$ has joined the multicast tree of $Server2$. Later, $Client6$ leaves the multicast group. $R6$ will notice that it is now forwarding over a single interface and send an adjust message down that interface. The adjust message will reach $Client7$ and $Client7$ will send a re-join message upstream. Now since $R6$ is no longer announcing anycast membership, the re-join request will be sent to $R5$ which is the closest anycast group member to $Client7$. Routers $R6$ and $R7$ are maintaining join state but both are waiting for the leave adjust timer to expire to remove the join state. Therefore these routers can not respond to the re-join request. In this case the re-join travels all the way to router $R5$. $R5$ generates an acknowledgement and sends it back down to $Client7$. Routers $R6$ and $R7$ both receive this acknowledgement and stop their leave adjust timer. Now $Client7$ has just rejoined to the tree of $Server2$ even though $Server1$ had a closer on-tree router, $R2$.

By reducing the number of routers joining the anycast group, the cost of anycast group membership announcements is reduced but so is the performance of this protocol. As the above example illustrated, leave adjustment may not perform as well since not all routers are announcing anycast membership. There can also be situations where a client joins the tree of a multicast server when another server would have provided a closer on-tree router on to which to graft a new branch. Our performance evaluation only considers the anycast-based protocol as it was originally described with the assumption of a GIA style architecture for global anycast groups.

5 Router-Supported Multicast Server Selection

We describe our router-supported multicast server selection protocol in this section. Note that this protocol is somewhat related to a previous proposal we have made for *receiver-based scoping* [25, 26]. That prior work, however, did not target multicast server replication. Our router-supported multicast server selection protocol allows the client to determine the grafting cost for each of the multicast servers. The client then chooses the server that has the lowest grafting cost. We envision the selection protocol being implemented in a selection module present in clients and servers.

5.1 Server Selection

The router-supported multicast server selection protocol works by having the replicated multicast servers periodically send out *cost* messages on a multicast control group, G_c. These cost messages are used by the selecting clients to determine the cost of the links that would be grafted if the client where to join the tree of the given multicast server. We call this number the *multicast tree graft cost*, TGC. As the cost messages flow down the multicast control group the value of the TGC field in the cost message is changed. Routers receiving the cost message will, for each interface on which it is forwarding the control group data:

1. Check if they are forwarding data for the data group and multicast server specified in the cost message over this interface.
2. If so, set TGC to zero.
3. Otherwise, increment TGC by the cost associated with the link.

Once the cost message is received by the client the TGC field will indicate the cost to the nearest on-tree multicast router for the specified multicast server. Thus the TGC value indicates the cost of the links that would be grafted if the client were to join the multicast tree of the specified server.

When the application issues a selection request, the client starts a timer and waits for cost messages from the servers. Reception of a cost message causes the selection module to:

1. Compare the reported TGC to the lowest TGC received.
2. If this TGC is lower, store the server IP address and corresponding TGC value.
3. Otherwise, discard the message.

Once the timer expires, the module returns to the client the IP address of the server that had the lowest TGC. The client can then join the multicast group of that server using normal IGMP messages.

5.2 Leave Adjustment

With this multicast server selection protocol it is easy to allow the client to perform leave adjustment. The departure of a client from the multicast group changes the sharing level on links in the multicast tree. So to handle leave adjustment we add a *multicast tree prune cost*, TPC, field to the cost message. This field indicates the cost of the links that are being used exclusively for delivery of the multicast data to the client and would be pruned if the client were to leave the multicast group. Upon reception of a cost message the routers update the TPC field as follows:

1. If the router is forwarding data over more than one interface, set TPC to zero.
2. Otherwise, for each interface on which control group data is forwarded, increment TPC by the cost associated with the link.

Now clients desiring to perform leave adjustments will specify this desire as part of their selection request. The selection module will then continue to listen to the multicast control group after returning the IP address of the selected server. The TPC that the client has with the selected server can then be compared to the TGC values that the client would incur if it were to switch to other servers. If the TGC for a different server is lower than the client's current TPC the selection module reports the new server address to the client application.

5.3 Example Operation

To illustrate how this scheme would work consider Figure 5 which shows two multicast servers with four multicast clients, $1-4$, receiving the multicast data. There are also two selecting clients, 5 and 6, listening to the multicast control group to receive cost messages from the two servers. The current group members are not interested in performing leave adjustment and thus did not remain members of the multicast control group. The TGC values that will be seen in the next cost message received by the selecting clients are given below the two clients. $Client5$ will join the multicast tree of $Server2$ since the TGC for $Server2$ is 5, and $Client6$ will join the multicast tree of $Server1$ since the TGC for $Server1$ is 6. $Client6$ is interested in performing leave adjustment and thus remains in the multicast control group after joining the data group of $Server1$. The cost messages received by $Client6$ after these joins will show a TPC for $Server1$ of 6 and a TGC for $Server2$ of 7. $Client6$'s TGC for $Server2$ has changed as a result of $Client5$ joining the multicast tree of $Server2$. Some time later, $Client1$ leaves

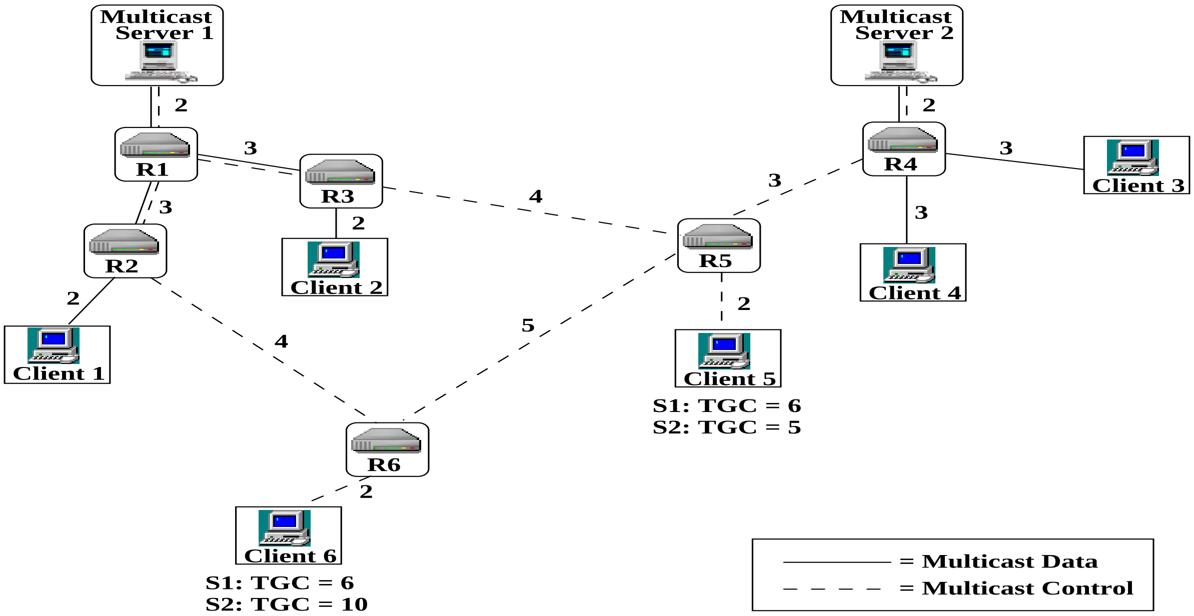


Figure 5: Router-Supported Multicast Server Selection Example

the multicast group. This means that the links from $R1$ to $R2$ are no longer shared. When $R2$ receives the next cost message from $Server1$ it will increment the TPC field to reflect this lack of sharing so that $Client6$ will see a TPC of 9. Since this is now higher than the TGC of 7 reported for $Server2$, $Client6$ decides to switch to the tree of $Server2$.

6 Performance

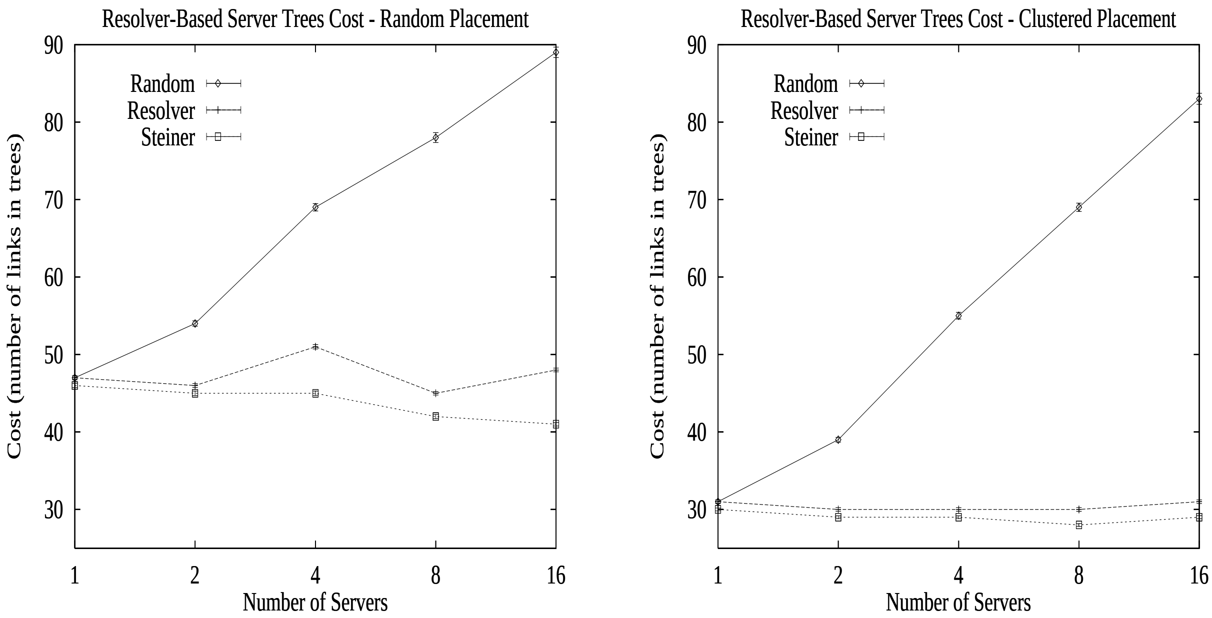
In this section we compare the performance of our three multicast server selection protocols. We present the results of a quantitative evaluation of the three protocols which was carried out using simulations. We used the Georgia Tech Internetwork Topology Models (GT-ITM) [27, 28] package to generate a 400 node graph consisting of 48 stub domains connected by 2 transit domains. Each transit and stub domain consists of approximately 8 nodes. We experiment with having 1 to 16 servers in a network with 400 nodes. We create event-driven simulations for each of the multicast server selection protocols. The number of multicast clients is set to be 5% of the 400 nodes. Clients are dynamic with an exponentially distributed mean time in group and mean time out of group set to 40 and 20 time units, respectively. We do not model packet losses and events that occur on a fine-grain time scale such as link delays. We experiment with random client placement as well as clustering the clients together. Source based trees are generated.

We evaluate the three multicast server selection protocols in terms of the hop count cost of the multicast server trees generated. We emphasize, however, that the protocols themselves can operate with various other metrics such as delay or available bandwidth. Hop count is a common measure and is practical in the sense that most network routing protocols use hop count as the routing metric. The network hop distance across which application data is transferred can affect the performance of applications. Various approaches have been taken to minimize the distance across which application data is transferred such as the replication of servers to store popular items close to a large group of clients [29, 30], and hierarchical caches that divide the network into hierarchical regions [31, 32].

In [16] the authors describe how to transform the multiple tree minimum cost allocation problem to a single tree routing (Steiner tree) problem. During the simulation, whenever a client joins or leaves, a Steiner tree approximation [33] is created and its cost is stored. This allows us to compare the costs of the trees generated by our multicast server selection protocols to the approximate Steiner tree cost. The tree cost displayed in the figures is the average over time for all of the server trees. A 90% confidence interval computed over 10 runs is shown in each of the performance figures.

6.1 Resolver-Based Results

We compare the performance of the resolver assignment to that received when random selection is used. The distance from the server to resolver is obtained by using the TTL field in IP packets. When the multicast servers send control group messages to a resolver, the initial TTL is specified and included in the data packet as part of the resolver-based multicast server selection protocol. When the packet is received by the resolver the number of hops traveled by the packet can be determined. This distance is then stored as the server-to-resolver distance. Figure 6 shows that the resolver-based multicast server selection protocol gives low cost trees for both random (Figure 6(a)) and clustered (Figure 6(b)) client placement. Figure 6(b) shows that with clustered client placement the cost of the trees is reduced compared to those with random client placement. The cost of the trees increases in Figure 6(a) for the resolver-based method when four and sixteen servers are available because there are more servers available which causes the resolvers to re-select the server to which to assign clients. Since the information the resolvers store is a resolver to server distance, a client may be assigned to a server that does not provide the lowest grafting cost for the client causing more links to be grafted to the



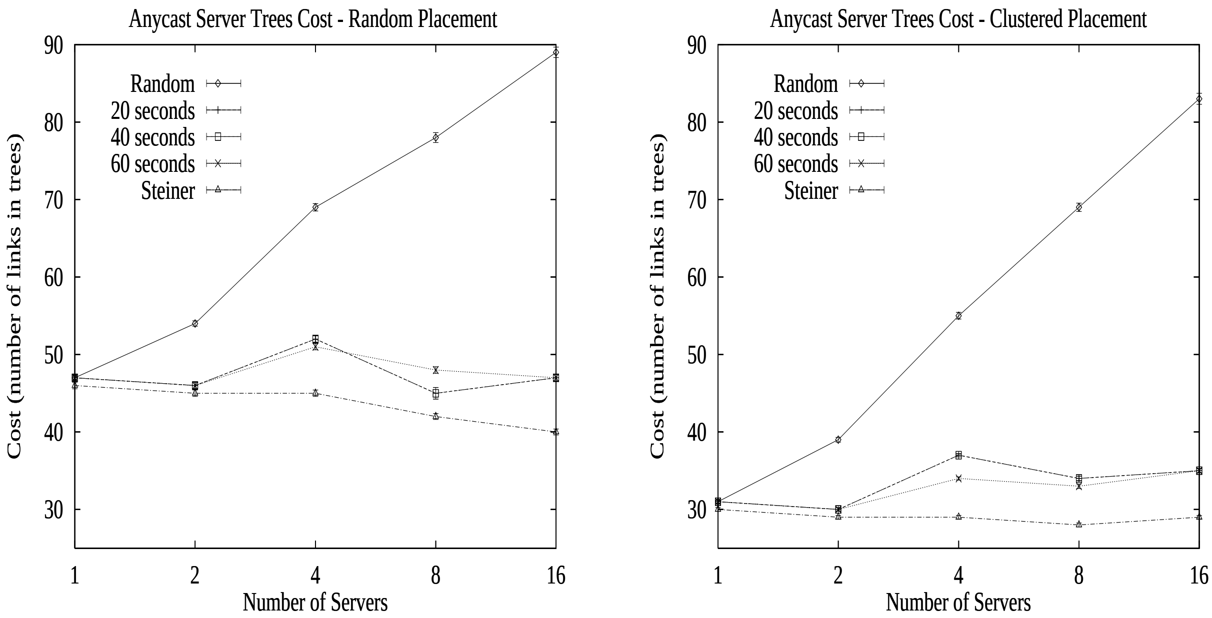
(a) (b)

Figure 6: Resolver-Based Results

tree. We see that when the clients are clustered together(Figure 6(b)) the algorithm performs better. The resolver-based protocol tree cost is close to the Steiner approximation suggesting that having the resolvers store more information would not give much reduction in the cost of the server trees. It seems that without information about the actual path from the server to client, using only the server distance to assign clients works well. Being able to determine the server to resolver distance to use in the multicast server selection gives much better performance than random selection of a server.

6.2 Anycast-Based Results

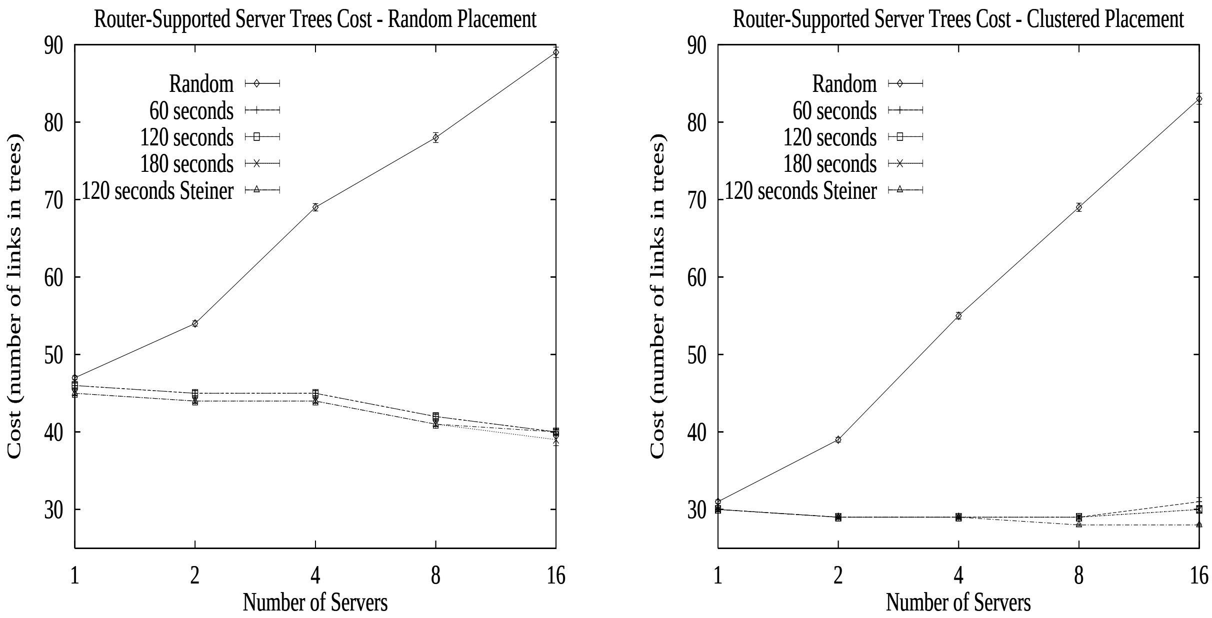
In the anycast-based simulations, a home domain is chosen randomly from the domains containing a multicast server and the anycast timer is varied from 20 to 60 seconds. The anycast timer is used by the routers to determine when to start announcing themselves as anycast group members. This timer controls the rate of anycast updates required within the domain. Figure 7 shows that varying the anycast timer had little effect on the cost of the trees generated. Only when the timer reaches 60 seconds is any change seen in the cost of the trees generated. The



(a) (b)

Figure 7: Anycast-Based Results

anycast timer has little effect on the tree cost because the border router will route the anycast packet requesting multicast membership using either a route towards the home domain of the anycast group or a cached route. Until the routers in the domain join the anycast group causing the BR's to mark the group as being internal, join requests from clients within the domain follow a route to the nearest anycast member. If the BR routes towards the home domain, the packet will be delivered to the nearest anycast member that is along the route and if a learned route is used it should also lead to the nearest anycast member. So the anycast timer allows a domain to control the rate of anycast updates which affects cost of the path grafted onto the multicast tree for new group members within that domain. Again the increase in tree cost when there are four and sixteen servers available is due to the change in server selection as a result of adding more servers. This is due to the fact that the protocol selects the servers based solely on the closest anycast group member. So the cost of the path grafted may be low but the overall cost from the client to the server may be high. Thus as clients join and leave unusually high cost paths may exist in the multicast trees.



(a) (b)

Figure 8: Router-Supported Results

6.3 Router-Supported Results

The performance of this server selection method depends upon the timing of cost messages. We, therefore, vary the message interval to determine the effect on the size of the multicast trees generated. The cost message interval also affects the time it takes the client to select a server. The longer the interval, the longer the client must wait before the server selection is completed, since the client will need to receive a cost message from each of the multicast servers. Figure 8 shows that as the message interval increases, the size of the trees generated slightly decreases. However, this decrease in the cost of the multicast trees comes at the cost of an increased join latency for the client. The client must wait for a message from a server to determine the server's IP address. So a longer message interval translates to increased join latency and decreased multicast control group overhead since the protocol messages are sent less frequently. The join latency can be reduced by having the clients specify how long they desire to wait before selecting a server. This way the selection will be only from the servers from which the client has received control group messages.

7 Concluding Remarks

We have described and evaluated three multicast server selection protocols targeting an application where the multicast servers are transmitting long-lived multimedia streams. In the resolver-based approach clients contact a resolver which responds with the multicast server to contact. With this approach, resolvers use selection cost metrics to assign clients to the servers. The second approach, an anycast-based approach, requires that on-tree routers join an anycast group and the clients send their request to join the group to that anycast address. Thus, the anycast service is used to find the nearest on-tree router to the client. The third server selection approach uses router-support so that the clients can make the decision on which server is "best". A client is given the cost of the links that would need to be grafted in order to join the tree of the specified server and the "best" server is the one with the lowest cost.

	Messaging	State	Leave Adjustment
Resolver-Based	Medium	Low	No
Anycast-Based	Low	Medium	Yes
Router-Supported	High	High	Yes

Table 1: Summary of Multicast Server Selection Protocols

Based on our quantitative evaluations, Table 1 provides a qualitative summary of the relative merits of the three protocols in terms of performance and cost. We see that the resolver-based approach has medium message cost with only the client query and periodic server to resolver messages being necessary. The state required is low since state is maintained only at the resolvers. The anycast-based protocol has a low message cost with messages being generated as a result of routers joining and leaving the anycast group. The anycast-based approach only requires state at the routers that are a part of the multicast group with extra state being generated due to the routers joining and leaving the anycast group. The overhead of this protocol can be further reduced by modifying the protocol to require only branching point and designated routers to join the anycast group. However, this reduction in protocol overhead comes at the cost of protocol performance in generating low cost trees. The router-supported protocol has a high messaging and state cost. This is because the replicated servers periodically send cost messages to the multicast control group and state is maintained in the network routers along the path of the control group. There is also an increased join latency with the router-supported approach since the client must wait for multiple messages before the selection can be made. Finally, the anycast-based and router-supported protocols are both able to handle leave adjustment, re-selection of a server due to members leaving the multicast group.

All three protocols are able to generate low cost trees that are much lower than the trees generated using random selection of a multicast server. If a receiver-based scoping or anycast service is available in the wide area network, using the router-supported or anycast-based multicast server selection protocols would be practical. However, deploying these services solely for the purpose of server selection is probably not reasonable. Of the three protocols, the resolver-based server selection protocol is the best protocol to be deployed in the Internet today. The resolver-based protocol results show that the use of resolver to server cost measures during the server selection gives much better performance than random selection. Our anycast-based protocol is a promising approach to be used when a global anycast service is provided in the Internet.

References

[1] S. E. Deering, "Host extensions for IP multicasting," RFC 1112, IETF, Aug. 1989.

[2] W. Fenner, "Internet group management protocol, version 2," RFC 2236, IETF, Nov. 1997.

[3] B. Cain, S. Deering, B. Fenner, I. Kouvelas, and A. Thyagarajan, "Internet group management protocol, version 3," Internet Draft, IETF, Mar. 2001. Work in progress.

[4] S. Deering, D. L. Estrin, D. Farinacci, V. Jacobson, C. Liu, and L. Wei, "The PIM architecture for wide-area multicast routing," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 153–162, Apr. 1996.

[5] D. Waitzman, C. Partridge, and S. E. Deering, "Distance vector multicast routing protocol," RFC (Experimental) 1075, IETF, Nov. 1988.

[6] 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.

[7] T. Ballardie, P. Francis, and J. Crowcroft, "Core based trees (CBT) an architecture for scalable multicast routing," in *Proceedings of ACM SIGCOMM '93*, pp. 85–95, Sept. 1993.

[8] R. Clark and M. Ammar, "Providing scalable web service using multicast communication," *Computer Networks and ISDN Systems Journal*, vol. 29, pp. 841–858, Aug. 1997.

[9] K. Almeroth, M. Ammar, and Z. Fei, "Scalable delivery of web pages using cyclic best-effort multicast," in *Proceedings of Infocom*, vol. 3, pp. 1214–1221, Mar./Apr. 1998.

[10] M. Ammar, K. Almeroth, R. Clark, and Z. Fei, "Multicast delivery of web pages or how to make web servers pushy," in *Proceedings of Workshop Internet Server Performance (WISP '98)*, June 1998.

[11] K. Almeroth and M. Ammar, "On the use of multicast delivery to provide a scalable and interactive video-on-demand service," *Journal on Selected Areas of Communication*, vol. 14, pp. 1110–1122, Aug. 1996.

[12] K. A. Hua and S. Sheu, "Skyscraper broadcasting: A new broadcasting scheme for metropolitan video-on-demand systems," in *Procedings of ACM SIGCOMM '97*, vol. 27, pp. 89–100, Oct. 1997.

[13] K. C. Almeroth and M. H. Ammar, "The interactive multimedia jukebox (IMJ): A new paradigm for the on-demand delivery of audio/video," in *Proc. of 7th International World Wide Web Conference*, vol. 30, pp. 431–441, Apr. 1998.

[14] L. Gao, J. Kurose, and D. Towsley, "Efficient schemes for broadcasting popular videos," in *Proceedings of NOSSDAV '98*, pp. 317–329, July 1998.

[15] S. Mahajan, M. Donahoo, S. Navathe, M. Ammar, and S. Malik, "Grouping techniques for update propagation in intermittently connected databases," in *Proceedings the 1998 IEEE Conference on Data Engineering*, pp. 46–53, Feb. 1998.

[16] Z. Fei, M. Ammar, and E. Zegura, "Optimal allocation of clients to replicated multicast servers," in *Proceedings of International Conference on Network Protocols*, pp. 69–76, Nov. 1999.

[17] C. Partridge, T. Mendez, and W. Milliken, "Host anycasting service," RFC 1546, IETF, Nov. 1993.

[18] R. L. Carter and M. E. Crovella, "Dynamic server selection using bandwidth probing in wide-area networks," Tech. Rep. BU-CS-96-007, Boston University, Boston, MA, 1996.

[19] S. Seshan, M. Stemm, and R. Katz, "SPAND: shared passive network performance discovery," in *Proceedings of USENIX Symposium on Internet Technologies and Systems*, pp. 135–146, Dec. 1997.

[20] S. G. Dykes, C. L. Jeffery, and K. A. Robbins, "An empirical evaluation of client-side server selection algorithms," in *Proceedings of Infocom*, vol. 3, pp. 1361–1370, Mar. 2000.

[21] E. W. Zegura, M. H. Ammar, Z. Fei, and S. Bhattacharjee, "Application-layer anycasting: a server selection architecture and use in a replicated web service," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 455–466, Aug. 2000.

[22] P. Francis, S. Jamin, V. Paxson, L. Zhang, D. Gryniewicz, and Y. Jin, "An architecture for a global Internet host distance estimation service," in *Proceedings of Infocom*, vol. 1, pp. 210–217, Mar. 1999.

[23] W. Fenner and S. Casner, "A "traceroute" facility for IP multicast," Internet Draft, IETF, Mar. 2000. Work in progress.

[24] D. Katabi and J. Wroclawski, "A framework for scalable global IP-anycast (GIA)," in *Proceedings of ACM SIGCOMM 2000*, vol. 30, pp. 3–15, Aug. 2000.

[25] G. Riley, M. Ammar, and L. Clay, "Receiver-based multicast scoping: A new cost-conscious join/leave paradigm," in *Proceedings of International Conference on Network Protocols*, pp. 254–261, Oct. 1998.

[26] L. M. Clay and M. H. Ammar, "Design and evaluation of router-supported and end-to-end multicast receiver-based scoping protocols," in *Proceedings of International Conference on Computer Communications and Networks*, pp. 384–389, Oct. 1999.

[27] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee, "How to model an internetwork," in *Proceedings of Infocom*, vol. 2, pp. 594–602, Mar. 1996.

[28] K. L. Calvert, M. B. Doar, and E. W. Zegura, "Modeling Internet topology," *IEEE Comunications Magazine*, vol. 35, pp. 160–169, June 1997.

[29] J. Gwertzman and M. Seltzer, "The case for geographical push caching," in *Proceedings of HotOS'95: The Fifth IEEE Workshop on Hot Topics in Operating Systems*, (Washington), May 1995.

[30] M. Rabinovich and A. Aggarwal, "RaDaR: a scalable architecture for a global web hosting service," *Computer Networks and ISDN Systems Journal*, vol. 31, pp. 1545–1561, May 1999.

[31] A. Chankhunthod, P. Danzig, C. Neerdaels, M. F. Schwartz, and K. J. Worrell, "A hierarchical internet object cache," in *USENIX 1996 Annual Technical Conference*, (San Diego, California), Jan. 1996.

[32] H. Che, Z. Wang, and Y. Tung, "Analysis and design of hierarchical web caching systems," in *Proceedings of Infocom*, (Anchorage, Alaska), Apr. 2001.

[33] A. Zelikovsky, "An 11/6-approximation algorithm for the network steiner problem," *Algorithmica*, vol. 9, pp. 463–470, May 1993.