An Extension of Scalable Global IP Anycasting for Load Balancing in the Internet

Siddhartha Saha, Kamalika Chaudhuri, and Dheeraj Sanghi

Department of Computer Science and Engineering
Indian Institute of Technology
Kanpur, UP - 208016, India
dheeraj@cse.iitk.ac.in

Abstract. IP anycasting was considered to be non-scalable, until Katabi *et al* [5] proposed a scheme called GIA. This scheme makes two important assumptions. One, the anycast addresses have a specific format distinguisable from unicast address. Two, it is only necessary to reach the nearest anycast member only if there is a lot of traffic to the anycast group.
In this paper, we propose an extension of GIA to achieve load balancing in the Internet. Our scheme builds upon GIA by adding load information exchange features. This enables the client to choose a member of the anycast group which is not just routing-wise nearest to it, but the one which actually improves its response time, by a proper balance of routing proximity and load. An advantage of our scheme is that it does on-demand routing information exchange, resulting in less overhead. We evaluate the performance of our design through simulation, and show that it achieves better load balancing, compared to GIA, which sends packets to the nearest server, irrespective of the load.

1 Introduction

Anycasting is a network service in which a data packet addressed to an anycast address can be sent to any member of a group of designated hosts, preferably the host that is nearest to the sender. Anycasting was first introduced by Partridge et al [6]. They define anycasting as a "stateless best-effort delivery of an anycast datagram to at least one host, and preferably only one host, which serves the anycast address." It is preferred that this host be the "nearest" from the sender, compared to all other members of the anycast group. Network can define "nearest" according to any routing metric - hop count, delays, etc. Anycast has been adopted by IPv6 as well [4, 1].

An important use of anycasting will be in providing a mirrored network service in a manner transparent to the user.The network will automatically select a server which is nearest to the client.

Another application can be a sort of auto-configuration [6]. Instead of configuring addresses of different servers for each machine, one may keep all the servers providing the same service in a well-known anycast group.

H.-K. Kahng (Ed.): ICOIN 2003, LNCS 2662, pp. 161–171, 2003.

Traditional approaches to anycasting either impose the restriction that all members of an anycast group should be confined within a single domain, or attempt to distribute information about all the members of each anycast group, to every network in the world. While the first one severely limits the utility of anycasting, the second one does not scale at all.

An interesting design for implementing scalable IP anycasting in the entire Internet was proposed by Katabi et al [5]. They have made two assumptions for this design. One, IP addresses used for anycast groups are of a particular format, which is distinguishable from unicast addresses. Two, it is not necessary to route each packet to the nearest member of the group. It suffices to route the packet to any one server in the group. Their design, Global IP Anycasting (GIA), provides default routes to all anycast groups (to its home domain). The address of home domain can be derived from the anycast address.

However, a plain implementation of GIA does not guarantee load balancing. GIA always tries to find the anycast host which is physically (or routing-wise) nearest to the client. This may not always be the best host in terms of response time for the client. This host may be overloaded, and it might be a better idea to contact the next-nearest server for a particular request. Our scheme extends GIA and provides load balancing features to the same.

Apart from GIA, there have been other schemes to do anycasting in the Internet [8, 2]. All these have examined issues related to the implementation of anycasting. However, none of them have a scalable solution for global anycast groups.

The remaining part of the paper is organized as follows. In Section 2, we describe the design and implementation of our scheme. In Section 4, we discuss simulation experiments, and provide performance results. Finally in Section 5, we conclude.

2 Extension for Load Balancing

The main principle behind our design is to provide an efficient scheme for the exchange of highly dynamic load information between the client and the anycast group members and using this load information decide on the optimum anycast server in such a way to minimize the overhead of load information exchange.

2.1 Design Principles

GIA works well with relatively static or slowly changing metrics such as hop count or routing delays. However, a somewhat different framework is required if we want to do routing based on highly dynamic metrics such as load information. What we need in this case is an efficient mechanism of load information exchange between the server and the client; this load information propagation should be scalable, should involve as little overhead as possible, and should preferably be propagated on-demand.

Our scheme has a fast load information exchange framework to take the load variations of the anycast servers into account. But any framework which adapts itself to a highly dynamic scenario, like change in the load information, has the potential of being unstable and can flood the network with routing data packets. We have avoided this by minimizing the additional routing information that needs to be exchanged between the routers.

Whenever a router conducts a search, it estimates the current load on its reachable servers. After the search is finished, the border router switches its anycast server only if it finds that its current anycast server is much more heavily loaded than most of the other anycast servers in the network. This approach helps reduce the unnecessary switches of anycast server and provides the required stability that we seek.

2.2 Framework

In our model, we assume that each server of the anycast domain resides in an autonomous network, which is connected to other autonomous networks, or, the Internet, through a border router. Each anycast group member estimates its own load periodically, and indicates its load information to its Border Router at regular intervals of time, or, when its load status changes by some threshold amount. The Border Router would in turn provide this load information to the clients who are getting service from that group member.

We call the Border Router associated with the anycast member as the Server Side Border Router and the Border Router associated with the edge domain as the Client Side Border Router.

Client side border routers initiate a search for a nearby member of the anycast group with low load when both the following two conditions are satisfied: One, when the border router has determined that this group is popular (that is, there are several packets going to this group). Two, when the latest load information about the anycast server that it has been in touch, is substantially higher than what was assumed earlier. Search mechanism is described in more detail later in Section 2.4.

The client-side border router would store the anycast routes found during the searches. When the border router has to send a packet to an anycast address which is popular for that domain, it tunnels the packet directly to the Border Router of the domain containing the chosen member of that anycast group. Each client domain also stores the load information of the chosen anycast member.

2.3 Load Information Exchange

As mentioned earlier, the server side will maintain its load information. This information will be sent out only when it is substantially different from the earlier information, or the previous information was sent a long time ago.

The information is not broadcast by the server-side routers, but instead, it is sent as unicast packets only to those client-side routers which need it. Server-side router needs to identify such client-side routers who have old information.

For this purpose, we maintain a sequence number at the server-side router. The number is incremented either when the load changes substantially, or when a timer expires. Whenever server-side router sends its load information, it includes this number in the packet.

We propose adding an extra field to the tunnel packet - the sequence number of the load information packet that it last received from the server. So, whenever a client-side border router tunnels a packet to the server-side border router, the latter knows the sequence number of the latest load information available to the client. If this sequence number is different from the current sequence number, only then the server-side border router sends back a load information packet to the client.

2.4 Search

The mechanism of search is essentially the same as that in GIA. The search is done by the border routers. The border router which initiates the search for a particular anycast address, sends a path query packet to its neighboring border routers, which in turn propagate the query to their neighbors, upto a predefined hop limit. If any of the border routers which received the query packet, knows a path to the anycast address in question, it sends a reply to the initiator router.

But we have made changes in the BGP [7] packets proposed in that scheme. In this case we need to propagate the load information also along with the search reply packet. Thus the BGP search reply packet is modified to accommodate the load information along with the sequence number.

One of the important consideration in designing the system is stability. If the load of one member of an anycast group increases, then a naive approach to load balancing would cause most of the client domains which were associated with that particular anycast member to initiate a search and possibly associate with another anycast member. This kind of behavior can easily lead to large oscillations in the load profile of the anycast members and instability in the network. By instability we mean that the client domains change their anycast servers frequently, thus leading to generation of more and more search packets and consequently, load information packets. In order to curb this oscillating behavior, we have introduced some dampening measures.

When the client-side border router receives a new load information packet from the server-side border router, it compares the current load information of the server with the average load of the anycast group. The average load of the anycast group is calculated as the average of the load values of all the servers for which the client domain received estimates during the previous search. This is admittedly older information, but it gives some idea about what kind of load is there on the group as a whole. And one would switch servers only if the server is more highly loaded than the rest of the group; otherwise we would do better by sticking to our current server by avoiding the overhead of searching.

After a client-side border router has initiated a search, it waits for a fixed time interval T for replies. Since some of the neighboring border routers may have a stale estimate of the load information of a server, we need each border

router to transmit the sequence number along with the load information value. If different estimates are received for the same server from two border routers, the more recent one is considered. After time T, the border router looks at all the responses received, and calculates the average load of the network. If the average load is still less than the current server load by a threshold, it has to choose a different server. To do so, it chooses k least-loaded servers out of all the responses received; out of these, it chooses the nearest. Our experiments show that with the current Internet topology, and 0.5 percent of the domains randomly containing an anycast server, a good value for k would be about 3. A larger value would result in simply choosing the nearest server disregarding any consideration for the load and smaller values would ignore any neighborhood information and would result in choosing simply the least loaded server.

If the current load information is higher than the average load value by a threshold (assumed to be 20% in our experiments), the client-side border router should try to switch servers. However, if all clients switch servers whenever the load at a server increases, the load balancing would become unstable. Here, we adopt a probabilistic approach in deciding whether a router should switch server when the load in the server is has increased by a substantial amount. Ideally, only those many routers should switch servers such that the load in the anycast server drops to an amount where its load becomes roughly equal to the average load in the network. We define the probability p_{switch} of initiating search for switching the server when the increase in load of the anycast server is over the threshold value as:

$$p_{switch} = \frac{load_{server} - load_{network}}{load_{server}} \ . \tag{1}$$

where $load_{server}$ is the current load of the anycast server and $load_{network}$ is the estimated average load of the network.

Note that, we keep the information of the average load of the network, hence, we do not need any extra information for the calculation of this probability measure. But there may be cases when these searches would not be sufficient to reduce the load in the anycast server, especially when there is a huge change in the load of the network. To protect against this, the routers which do not fire search immediately, waits for an interval of time T_{max} before scheduling the search. T_{max} is estimated to be roughly the time period needed to complete the search. In this period, some clients may have switched their servers, and the load of the overloaded server may have decreased as a result. But if after the time T_{max}, the load of the server still remains greater than the average load of the group by a threshold, the same algorithm is followed; otherwise no action is taken.

2.5 Load Metric

Our method allows an anycast group to use a load metric that is relevant to the application; the only constraint is that the same metric should be used for all members belonging to an anycast group. In our experiments, we had assigned

each server a capacity value, which is the maximum number of packets that it can server in an unit time. The load was defined as the ratio of number of packets served in an unit time to this capacity.

3 Experiment

To evaluate the performance of our scheme, and compare it with GIA, we have conducted simulation experiments. We describe below the simulation setup, parameters, and the results obtained.

3.1 Simulator

We implemented a custom simulator to study the performance of our scheme. For the simulation topology, we use a set of snapshots of the Internet inter-domain topology generated by NLANR based on the BGP routing tables [3].

In our experiments, we simulated only one anycast group. This is because, in our scheme, load balancing for each anycast group is independent of the other anycast groups in the Internet. By having only one representative anycast group, we could increase the number of domains, number of members of the anycast group, etc., and run simulations for longer periods of time.

Given a graph representing the Internet domain topology, we randomly assign n domains to have members of the anycast group. Out of the remaining domains, a fraction p of the domains consider that anycast group as a popular group. For the rest of the nodes, the anycast group may be accessed only rarely. One of the domains having anycast group member is randomly marked as the home domain.

As the simulation starts, a series of requests are generated by each domain, and these requests are routed to the appropriate anycast group member, chosen according to the scheme we have proposed. The desired quantities, namely load profiles and processing delays are then measured at each anycast group member. Each node starts sending requests after a random time interval, to avoid synchronization of requests.

3.2 Parameters

To measure the performance of our scheme for various load conditions, we define several parameters. We define C, the capacity of an anycast server, as the number of packets it can process in unit time. We used a simplistic queue management algorithm, in which if an host receives more that C packets in an unit time interval, the extra packets are dropped, and the originator node of that packet is notified. We also keep track of the number of packets dropped in this manner during the simulation period.

In order to compare the performance of our extensions, with the original GIA, we use three parameters. We compute average queuing delays at the server

side. When a packet arrives at the server, the number of packets in the queue multiplied by the service time of a single packet, gives the queuing delay.

Our second parameter is the path length (hop count) from the client domain to the chosen anycast server. Finally we measure the load profiles at the anycast hosts. To get a measure of the average load condition of the network, we define another parameter L. L is given as

$$L = \frac{\frac{1}{2}(n_p p + n_u(1-p)) * N}{n * C} \ . \tag{2}$$

Here p is the fraction of domains which have the anycast group as a popular group, n_p is the maximum number of packets that a client can send to a popular group in unit time, and n_u is the maximum number of packets that a client can send to other groups in unit time. The actual number of requests generated by a source (client) is a uniformly distributed random number between 0 and n_p or n_u (as the case may be). N denotes the total number of nodes in the graph, and n is the number of anycast servers. Number of domains in the Internet (N) is taken as 6474 [3]. The numerator indicates the expected number of packets generated in the network per unit time and the denominator is the total number of packets that can be processed by all the servers in unit time. The ratio thus gives a measure of the overall load in the network.

The other parameters that we use in our simulations is n, the number of domains containing an anycast host and p, the fraction of domains for which the anycast group is popular. Simulations were performed under two different load conditions (medium load, $L = 0.25$ and high load, $L = 0.55$) for different values of n and p.

3.3 Results

We wanted to measure the performance of this load balancing scheme under various scenarios, by varying different parameters of the network model. We performed two sets of experiments. In the first set, we studied the performance of our scheme with increasing values of n, the number of members in the anycast group. n is varied between 20 to 140. In these experiments, it is assumed that 60% of the domains consider the group as popular ($p = 0.6$).

In the second set of experiments, we study the performance of our scheme as p, the fraction of domains for which the anycast group is popular, increases. The value of p is varied between 0.1 to 0.9. It is assumed that the anycast group has members in 60 domains ($n = 60$).

For both sets, we perform simulations for both medium and high load conditions,viz. $L = 0.25$ and $L = 0.55$, for GIA and our scheme. The simulation results are described below.

Load Balancing Figure 1 shows the variance of the loads at the anycast servers for GIA and our scheme for two different load conditions in the network, as a function of number of members in the anycast group.

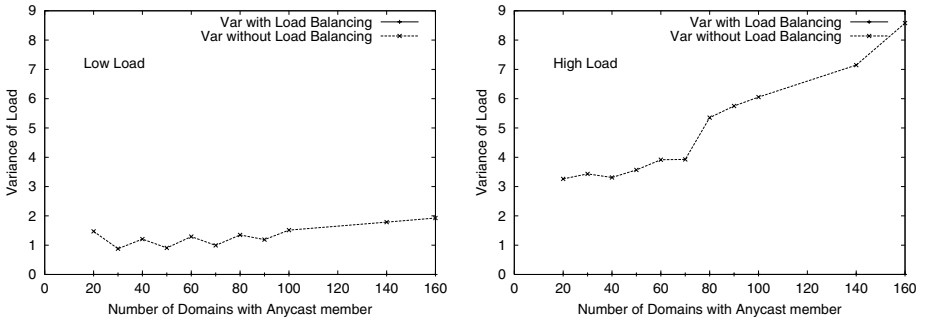

Fig. 1. Variance of the load (with and without load-balancing) among the anycast group members, under low-load and high-load situations, as a function of the number of anycast servers in the network

We notice that when load balancing is implemented, the variance in loads of different servers is always close to 0, irrespective of load as well as the number of members in the anycast group.

This shows that our scheme is actually able to do balance the load between the servers rather well.

Our experiments with varying percentage of the domain which have this anycast group as popular also show same kind of result. i.e., close to zero variation with the load balancing scheme enabled.

Queueing Delays In the case where load-balancing is not done, we find that some servers don't receive any requests, while some other servers have huge queues built up, and the average queue-size is quite high. In case of load-balancing, all servers get roughly equal number of requests, and hardly any queues are built anywhere in the network.

Figure 2 shows the average queueing delays at the anycast servers for GIA and our scheme for two different load conditions in the network, as a function of the number of anycast servers.

As is expected, proposed extension is able to reduce average queueing delays to almost negligible values even under high load.

Figure 3 shows the average queueing delay at the anycast servers, with and without load balancing extension, for two different load conditions in the network, as a function of percentage of domains which consider this group as popular. Again the queueing delays are almost non-existent with load-balancing. Smaller queueing delays are obviously preferable, since it implies smaller response time for the client. Thus, load-balancing results in faster response to the client.

Path Length Since our scheme does not choose the nearest server, the average path length with the load balancing enabled may be more than the pure imple-

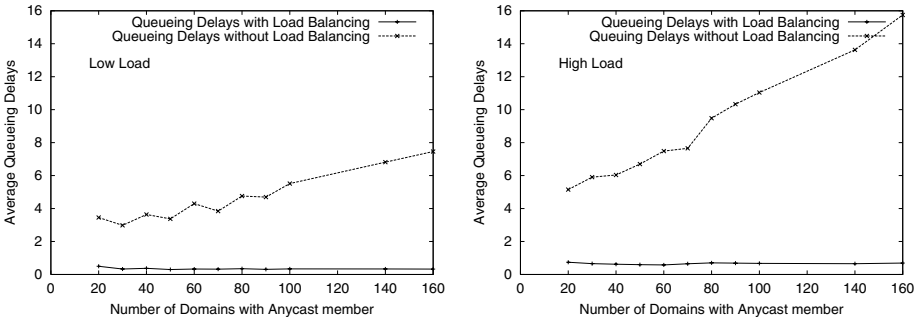

Fig. 2. Average queue size (with and without load-balancing) at the anycast servers, under low-load and high-load situations, as a function of the number of anycast servers in the network

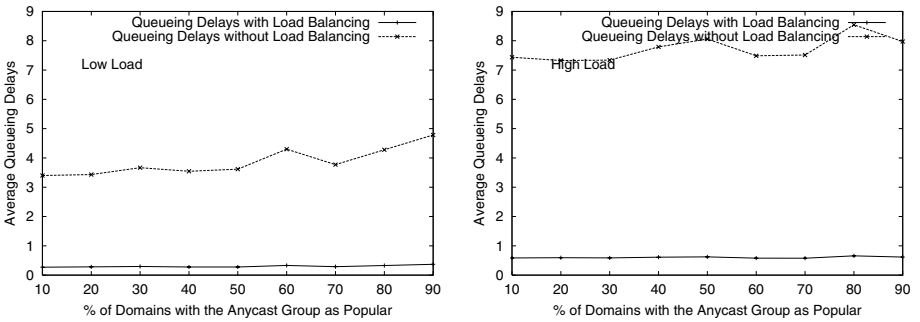

Fig. 3. Average queue size (with and without load-balancing) at the anycast servers, under low-load and high-load situations, as a function of the percentage of domains that consider this group as popular

mentation of GIA. To check this out, we looked at the average path length in case of both GIA and load-balancing extensions.

Figure 4 shows the average path length (number of hops) for GIA and with load-balancing extensions for two different load conditions in the network, as a function of number of members in the anycast group. The average is taken over all clients which consider this anycast group as a popular group.

From the figure, we notice that there is a little increase in path length (one extra hop on average) with the load balancing schemes, as expected. The increase in round-trip time due to this extra hop will be more than compensated by smaller queueing delays and the much less number of dropped packets.

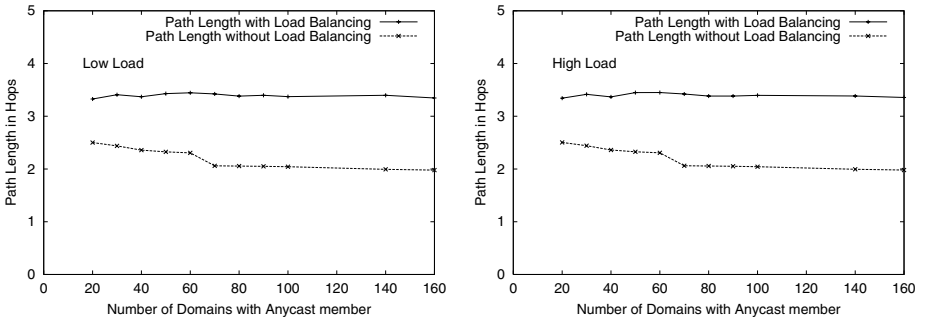

Fig. 4. Average path length (with and without load-balancing), under low-load and high-load situations, as a function of the number of anycast servers in the network

4 Conclusion

Our scheme builds upon GIA [5] by adding features which enable the exchange of dynamic load information. The information exchange happens only on demand, thereby not imposing large overhead on the network. Furthermore, we make sure that a border router does not switch servers until it is absolutely necessary, i.e., when the server load goes significantly higher than other anycast members, thus providing stability in the network. The simulation results show that the scheme is able to distribute the load on the servers evenly, under different load conditions. As a consequence of this, the queuing delays at the servers are much less for our scheme than for GIA. Finally, the average path length to reach an anycast server when using our scheme, is only higher by one hop on an average as compared to GIA.

References

[1] S. Deering and R. Hinden. Internet protocol version 6 (IPv6) specifications. RFC 2460, Dec. 1998. 161

[2] R. Engel, V. Peris, D. Saha, E. Basturk, and R. Haas. Using ip anycast for load distribution and server location. In *Proc. of Third Global Internet Mini-conference*, Nov. 1998. 162

[3] T. N. L. for Applied Network Research (NLANR). http://www/moat.nlanr.net/AS/. 166, 167

[4] R. Hinden and S. Deering. IP version 6 addressing architecture. RFC 2373, July 1998. 161

[5] D. Katabi and J. Wroclawski. A framework for scalable global ip-anycast (GIA). In *Proc. of ACM SIGCOMM 2000*, pages 3–15, Stockholm, Sweden, Aug. 2000. 161, 162, 170

[6] C. Partridge, T. Mendez, and W. Milliken. Host anycasting service. RFC 1546, Nov. 1993. 161

[7] Y. Rekhtar and T. Li. A Border Gateway Protocol 4 (BGP - 4). RFC 1771, Mar. 1995. 164

[8] D. Xuan, W. Jia, W. Zhao, and H. Zhu. A routing protocol for anycast messages. *IEEE Transactions on Parallel and Distributed Systems*, 11(6), 2000. 162