Using Network Layer Anycast for Load Distribution in the Internet

E. Basturk R. Engel R. Haas D. Kandlur V. Peris D. Saha

‡IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{basturk,rengel,haas,kandlur,vperis,debanjan}@watson.ibm.com
Phone: (914) 784-6625 Fax: (914) 784-6205

Abstract

In the Internet, when a unicast IP address is shared by many hosts, it is known as an anycast address. In contrast to multicast, a packet destined to an anycast address is forwarded to any one member of the anycast group. In this paper, we investigate how the IP anycast service can be exploited by hosts connected to the Internet without significantly impacting the routing and protocol processing infrastructure already in place. In particular, we study how routers running RIP and OSPF manage routes and forward packets to anycast destinations. We propose possible enhancements to routing and forwarding to fully exploit the potential of the anycast service. We also discuss changes required in the host protocol stack which would enable applications to transparently use the anycast service. More specifically, we propose a scheme that limits the required changes to the IP layer processing and describe its implementation in the AIX TCP/IP stack. Finally we consider an application example where anycast communication is used to distribute load among a set of mirror web sites. Using traces from the IBM Olympic web site we compare several different load distribution schemes, both in terms of number of connections served as well as number of bytes transferred.

Keywords: anycast, load distribution, Internet, routing, world-wide web

1 Introduction

In the Internet environment there are several instances where clients need to locate services provided by the network. In order to increase service availability and provide load distribution, it is now common practice to replicate servers that are providing these services. Examples include WWW "mirror" sites, multiple SOCKS servers [21] and proxy servers. These replicated services can be classified into (1) local replicas, and (2) distributed replicas. Local replicas, such as those provided on a cluster of workstations, may be contained within a single subnetwork. On the other hand, distributed replicas could be geographically dispersed across the Internet. For clients to be able to locate the appropriate server it is necessary to extend and enhance the location mechanisms used for directing clients to these services.

In the Internet it is fairly typical for a server to be identified by a name as opposed to a network address. Naming mechanisms such as the Domain Name System (DNS) [23] provide mappings from host

names to their IP addresses. DNS can been used to map virtual host names (service names) to a set of IP addresses that correspond to the servers providing the replicated service [6]. Take the case of a client making a Web access with a single name that is associated with a set of HTTP servers. When the client sends a name resolution query the DNS server maps the name to any one HTTP server. Each successive query can be mapped to an IP address associated with a different HTTP server in a Round Robin manner. This will distribute the client requests to the different HTTP servers and share the load among the servers. Rather than a simple Round Robin scheme the resolution process may also be guided by network and server load. The Distributed Director [8] as well as the proposed Host Proximity Service (HOPS) [14] direct the client to the "closest" server, where distance is measured with respect to some metric based on network topology. Along the same lines [4] proposes an application layer anycast service to support replicated services. Both Distributed Director and HOPS servers interact with the routing protocols to pre-compute a measure of the distance of the client from each of the candidate servers. While these approaches have some attractive features, they disable the caching mechanism of DNS and therefore are likely to increase the DNS query load. Past studies have shown that DNS related traffic is not insignificant [9]. Moreover, in order to provide appropriate responses for distributed replicas, the name server has to obtain fairly accurate network topology information.

An alternative approach to supporting replicated services is with the use of anycast addresses. An anycast address is an IP address that may be bound to one or more network endpoints. Different servers that are providing the same service can all have the same anycast address. A host is considered a member of an anycast group simply by virtue of the fact that the anycast address is bound to one of its interfaces. It is not necessary for any router in the network to be explicitly aware of the membership of an anycast group. A packet addressed to an anycast address is delivered to one of the members of the anycast group. This approach is elegant in that forwarding of packets bound to anycast addresses is handled directly by the routers which have access to accurate network topology information.

Anycast addresses can be used to efficiently distribute load across the different links in the network. We outline some simple policies that can be implemented in routers to effectively provide load distribution among different servers. With the help of traces gathered from the IBM Olympic Web site [2], we demonstrate how these simple policies fare in the case of distributing a large number of web accesses across a few web servers.

Traditionally, the IP service is only limited to delivering datagrams to end-hosts without considering how the applications use them. In this regard the anycast service is no exception. However most if not all applications have some notion of state that links successive packets together. Clearly, if as a result of anycasting, successive packets are forwarded to different anycast servers this state information will be lost, rendering anycasting useless for all but a few instances that involve a single packet exchange. In order to ensure that anycast datagrams belonging to a single flow are not routed to different end-hosts we need to make some minor modifications in the host TCP/IP protocol stack. The basic idea is to pin the end-host to which the first packet of the flow has been sent. This is very similar to route pinning in the context of QoS routing [15]. The pinning is done by inserting a loose source route option in all subsequent packets from the same flow. We have implemented these modifications in the TCP/IP protocol stack of AIX 4.2.

In this paper we explore some techniques for providing anycast services in the Internet. In Section 2 we briefly define the anycast service that is part of the Internet standard. In the subsequent section we outline

some of the implications of anycasting to the routing infrastructure that is currently deployed. Section 4 is devoted to the issue of ensuring that packets from the same flow are always routed to the same end host. There we describe both the transmit and receive modifications to the TCP/IP stack in AIX to ensure that stateful connections can be maintained between two end hosts. In Section 5 we describe some simulation results on the load balancing capability afforded by the use of anycast at the network layer. This is based on a simulation from a trace gathered from a heavily loaded web server. We conclude in Section 6 with a brief discussion.

2 Anycast Communication in the Internet

The notion of anycasting in the Internet was first introduced by an RFC produced by the Internet Research Task Force [26]. The authors of [26] motivated the need for an anycast address by giving examples of clients trying to locate a server that is closest to them. They define IP anycasting as: "a service which provides a stateless best effort delivery of an anycast datagram to at least one host, and preferrably only one host, which serves the anycast address". Anycasting is also defined in IP version 6 (IPv6) albeit with some minor differences [11]. In the following we clarify some of the issues that are part of the anycast service definition, namely: (1) identification of an anycast datagram, (2) forwarding of an anycast datagram, and (3) advertisement of anycast servers.

In both IPv4 and IPv6 an anycast datagram is no different from a regular IP datagram. The destination address field in the datagram corresponds to an anycast address. In [26] the authors suggest carving out the IP address space for a separate class of anycast addresses. They contend that this will reduce the risk of applications mistakenly failing to recognize anycast addresses. Apart from chewing up a significant amount of the IP address space which is already a scarce resource, this approach places an additional burden on the routing protocols to distribute reachability information related to these new addresses. The special anycast addresses will not aggregate with the regular unicast routing information and so it might result in excessive amount of route advertisements. In contrast, IPv6 specifies that anycast addresses are identical to unicast addresses, with the difference that it allows multiple hosts (more specifically, network interfaces) to be registered with the same address.

A router that receives an anycast datagram might have multiple next hops over which it can forward this datagram. It is recommended that the router picks any one of these possible next hops to forward this datagram. Since it is possible for unicast IP packets to be duplicated, it is possible for an anycast datagram to be delivered to multiple hosts. However, the specifications [11, 26] make it clear that the intended behavior is for anycast datagrams to be delivered to a single host.

Clearly, members of an anycast group have to indicate to the routers that they wish to receive anycast datagrams for a specific address. One approach is to use an enhanced version of IGMP so that hosts can advertise this information to routers. This will be particularly useful on shared media LANs so that all the routers on the same media will get this multicast information. Each router can then advertise its reachability to a host with an anycast address, through regular routing updates with its neighbors. Another approach is to have the end hosts run a routing protocol so that they can directly update the routers with the information about which anycast addresses they serve. Note that the hosts do not need to run the full blown routing protocol; rather they only need to be able to generate route advertisements. A third and

simpler approach is to statically configure the routers with the information about the different hosts that are serving an anycast address. In the next section we discuss some of these routing and forwarding issues with respect to some commonly used routing protocols.

3 Implications to Routing and Forwarding

In this section we explore the implication of anycast communication on routing and forwarding. Since the notion of anycasting presumes that there are possibly multiple end hosts with the same anycast address, we need to consider how this might affect the existing routing protocols. In this context, we examine two common routing protocols, RIP (Routing Information Protocol) [17, 22] and OSPF (Open Shortest Path First) [24] and discuss how they can be used to propagate the information about routes to several different end hosts that are sharing an anycast address. We use the term multipath route to denote the fact that a router has multiple routes to the same destination.

If a router has an equal cost multipath route, it has to choose a single route to forward the packet to the next hop. While anycasting, by itself, does not require any special forwarding behavior, supporting stateful connections to anycast destinations does pose some interesting challenges. We discuss a few different mechanisms that can be used to spread the packets out on the different routes achieving a fair amount of load-balancing. Additionally, one may have several hosts with the same anycast address residing on a single shared media LAN. In this case, it is the link-layer address resolution mechanism that has to recognize the presence of different anycast servers and consistently distribute anycast datagrams to each of them. We outline a solution for both the IPv4 Address Resolution Protocol (ARP) [28], and the IPv6 Neighbor Discovery Protocol (IPv6 ND) [25].

3.1 Routing Requirements

Since anycast addresses are no different from unicast addresses it is commonly believed that anycasting does not require any changes to the routing protocols. But having multiple hosts with the same IP (anycast) address may be a problem for the routing protocols. We examine this in the context of RIP and OSPF as they are representative of two distinct families of routing protocols namely distance-vector and link-state protocols.

Distance-vector routing algorithms are based on a distributed shortest path computation as described by Ford-Fulkerson [20]. The basic idea is that each router advertises all the IP destinations (or subnets) that it can reach with an associated cost, to all its neighbors. Each of its neighbors then recomputes its routing database to include information that it has just learnt and broadcasts this update on all its links. This process is repeated every time there is an update or at a periodic interval.

With these algorithms, if several nodes with the same anycast address are present in the network, then each node in the network will ultimately have the distance and next-hop address to their closest anycast node. As an example, consider the network shown in Figure 1. The nodes that share an anycast address are labeled as A and are shaded in the figure. The table in Figure 1 shows the cost as well as the next-hop information that will be computed at node S as a result of running RIP using the link costs shown in the figure. Note that node S has an equal cost multipath route to A, with the next hops being either B or C.

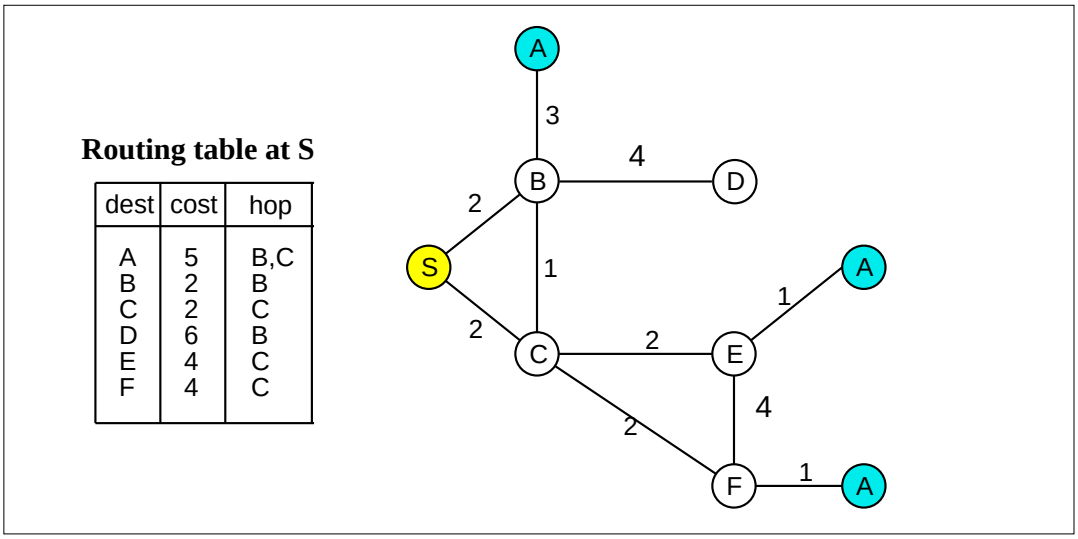


Figure 1: Example network with Routing Table at node S

Current implementations of RIP only keep one entry per destination in their routing table. In case an equal-cost route advertisement is received through another next-hop, it does not replaces the existing route, unless of course that route previously timed out. An optional heuristic used in the BSD 4.3 version of RIP, allows an update to a route to occur if it is close to timing out, otherwise the advertisement is dropped. For a meaningful use of the anycast service, it is imperative for routing protocols to support multipath routing. Keeping a single route among equal-cost routes is not a requirement of the RIP protocol, and if routers running RIP are to support the anycast service it is desirable that they have the capability to maintain multiple routing entries per destination.

Note that for distance vector protocols we only recommend multipath routing among equal cost routes. The reason for this restriction is the fact that spreading packets across routes which do not have the same cost is likely to result in loops. For example in Figure 1, node E has a route to A through C with a cost of 6. However, node C also has a route to A through E with a cost of 3. Thus, if E is spreading its packets among all the multiple routes that it has to A, it will eventually send a packet with destination A to node C. This packet is likely to be sent back to E by node C, resulting in a loop.

One can enhance the structure of the metric used by RIP (hop-count) to include bandwidth, delay, channel occupancy and reliability. This is done by a more sophisticated distance-vector routing protocol called IGRP (Inter-Gateway Routing Protocol [18]). IGRP allows packets to be forwarded among multiple equal-cost paths. Two paths are considered equal-cost if their composite metric is equal, where composite metric is defined as a combination of the various components of the metric. Originally IGRP also allowed the forwarding of packets between almost equal-cost paths, but the difficulty to prevent packets looping on such paths led to the disabling of this feature.

Link-State Routing Protocols assume that each node in the network has a complete picture of the topology of the network. This is achieved by each node flooding information about its links to all other nodes in the network. Every node that is participating in the routing protocol builds a topology database

from all the link updates it has received. Using this topology database it can compute the shortest path to any destination. If all the nodes in the network have the same topology database they will all choose the same path to the destination.

OSPF is a Link-State Routing protocol. Consider the example shown in Figure 1 and assume that all the nodes including the ones with the anycast address A are running OSPF. An interior node, say C will not be able to distinguish between the link state updates that are being sent by the different nodes with the anycast address A. Therefore it is not advisable to use regular link updates to advertise the availability of a host interface with an anycast address. Instead, one can define external routes to the anycast address on the router to which the host is attached. In case the host is running the OSPF routing protocol then the external route can be defined on the host itself. The metrics for these routes should be chosen to be of Type 1, i.e. metrics are expressed in the same units as OSPF interface cost. By including anycast routes as external routes in the network topology, these routes are not taken into account when the shortest path tree is built over the network topology.

OSPF leaves the handling of equal-cost multipath forwarding as implementation specific. Some implementations might use all equal-cost paths available, whereas others might only pick one of them. In general, any of the mechanisms described in the next section can be used to forward anycast datagrams. Since the complete network topology is known by each router running the OSPF protocol, it is possible to do multipath routing on paths which are not necessarily equal-cost. However, when doing multipath routing over unequal cost paths, it is important that all routers within an OSPF area are operating in a consistent manner to prevent the occurrence of loops.

The next section describes a few simple policies that can be used to distribute traffic across multiple equal-cost paths. If the objective is to balance the load equally among different servers then existing routing protocols do not provide enough information for this purpose. This is evident in Figure 1 where the only information available at node S is that it has several equal-cost paths over which it can reach A. The fact that the link from S to C can actually reach two servers that have the address A, as opposed to the link from S to B which can only reach one server is not available. This problem can be alleviated if in addition to the reachability information, the routing protocols advertise the number of hosts that are reachable. For instance in Figure 1, node C will advertise to S that it can reach 2 copies of A with cost 3, and node B will advertise to S that it can reach 1 copy of A with cost 3. This additional information will enable S to distribute the packets that are destined to A over the links $S - C$ and $S - B$ in the ratio of 2:1.

3.2 Forwarding Anycast Datagrams

One of our main objectives in this paper is to use the anycasting mechanism to effectively distribute traffic to several different servers. There are many schemes that are currently proposed for this purpose with differing capabilities as described in Section 5. One advantage of using anycast addresses for replicated servers is that load distribution is done naturally with respect to the routing topology. Therefore clients are automatically directed to the closest server (in a network topology sense).

In addition to directing clients to the nearest server, we would like to use the anycasting mechanism to balance the load between servers. To simplify the discussion let us first consider the case where multiple

servers sharing the same anycast address are directly attached to a single router. From a routing perspective these multiple next hops appear as a multipath route to a single destination. Now, given that the router has a multipath route to a particular address – the router cannot distinguish between unicast and anycast addresses – what is an appropriate way to distribute traffic among these multiple routes? Broadly speaking there are two ways in which this can be done: (1) Flow basis, and (2) Packet basis. The idea behind flow-based forwarding is to identify the flow that each packet belongs to, so that packets from the same flow can be directed to the same end-host. This method requires the router to keep track of the active flows that are passing through it. On the other hand, packet-based forwarding does not require any state to be maintained at the router; the router simply forwards each packet independently.

The *Connection Router* approach described in [12] is a good example of a flow-based forwarding scheme. A Connection Router maintains a list of all the active flows, typically TCP connections, that are passing through it. When an IP packet has to be forwarded, the Connection Router first checks to see if the protocol type indicates that it is TCP. If not, it just forwards the packet on any one of the alternative routes to its destination. Otherwise, it proceeds to search its list for the corresponding flow (identified by the source address/port number and destination address/port number). A successful match yields the next IP hop (route) that the packet has to be sent to. A packet belonging to a new TCP flow will fail this search, resulting in this TCP flow to be added to the list along with the associated next hop information. The determination of which next hop to choose for a particular flow is a local policy decision.

Note that this mechanism requires the router to maintain a significant amount of state information for all the flows that are being routed through it. Routers typically are optimized for forwarding packets. Modification to maintain flow states and classification of the incoming IP packets into flows is likely to impact the efficiency of the routers. Also, this mechanism is mainly suitable for cases where the replicated servers are topologically close to the Connection Router.

In the case of packet-based forwarding the router treats each packet on an individual basis. It forwards each packet using one of the possible routes without regard to maintaining consistency between the successive packets of the same flow. This requires the router to maintain absolutely no state as far as the flows are concerned. We consider three schemes for forwarding anycast datagrams: (1) round-robin, (2) random, and (3) hash-based. Assume that a router has an equal-cost multipath route to a certain destination. If the round-robin scheme is used, the router forwards successive packets addressed to that destination in a round-robin manner. With the random scheme, the router randomly picks one of the multiple routes to forward the packet. Note that both of these schemes can result in successive packets from the same flow being forwarded to different next hops. Consequently, anycast packets belonging to the same flow might end up at different end hosts, which is clearly not of much use to the application, especially when most IP traffic is "flow-based", e.g. telnet, HTTP, SMTP, etc. In Section 4, we describe modifications to the host protocol processing that alleviates this problem by adding a source route option in the packet in order to pin an anycast flow to a unique destination.

It is not necessary to maintain a list of active flows to ensure that the router forwards packets belonging to the same flow in a consistent manner. One can use a hash-based approach to consistently route all the packets in a flow to the same host. The basic idea here is to use the source address/port number in the packet as a key to selecting one of the multiple next hop candidates for the same anycast address. Assuming that there is no routing transient, this ensures that packets from the same flow are sent out on the same

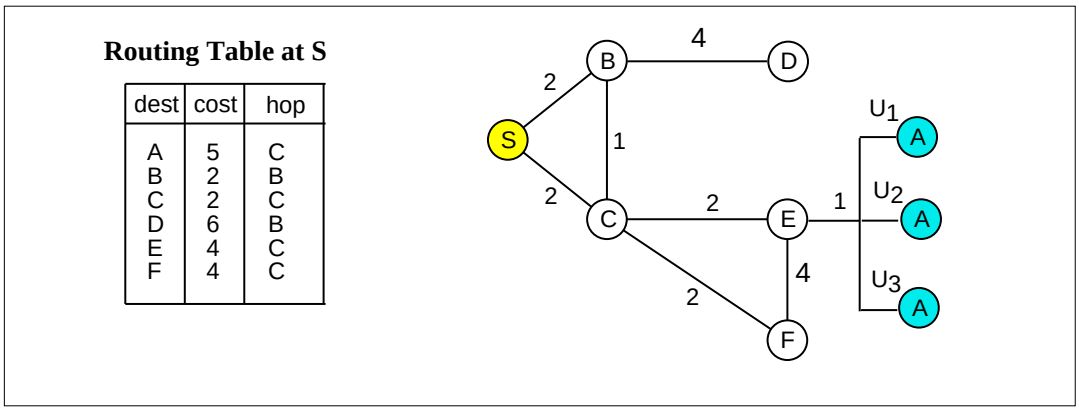


Figure 2: Example network with anycast servers on the same LAN

next hop at each of the routers along the way. Thus, all the packets of a flow are routed to the same end host. Note that this solution requires all the routers in the Internet to consistently forward packets belonging to the same flow to the same next hop. Another caveat is that a change in the multipath route at a particular router may lead to a change in the outcome of the hash function. Consequently, packets of the same flow may take different paths. This could result in a disruption of all the flows that share a multipath route. Typically, route changes do not occur very frequently in the Internet and so this disruption can be assumed to be fairly minimal, especially for short-lived connections [27, 7]

In Section 5 we examine the load-balancing performance of each of these schemes with the help of a trace-based simulation, with traces collected from the 1996 Olympic Web server.

3.3 Link-layer address resolution mechanisms

In this section we focus on how layer 2 mechanisms like IPv6 Neighbor Discovery [25] or IPv4 ARP [28] can be used to reach anycast destinations. This is especially useful when multiple anycast servers with the same anycast address are connected to a shared medium like Ethernet, as shown in Figure 2.

Consider the case where the router E is trying to forward a datagram that is destined to A. In the case of IPv6 Neighbor Discovery, the router E sends a Neighbor Solicitation message to request the link-layer address of A. All three nodes with address A answer with a Neighbor Advertisement message in response to the solicitation. The router at node E keeps the first Neighbor Advertisement message that it receives in response to its request. Subsequent Neighbor Advertisements received in response to this request are dropped by E. Thus the datagram (and all subsequent datagrams) destined to A will be sent out to a single interface. If the anycast server that can be reached on this interface happens to go down the Neighbor Unreachability detection mechanism in IPv6 will detect that the server went down and send out a new Neighbor Solicitation message. This time only the two remaining servers will respond and the first one to respond will be chosen. Thus there is a certain amount of fault tolerance that is given by using the same anycast address for all the three servers.

In the case of IPv4 ARP, the main difference is that there is no Neighbor Unreachability Detection

mechanism. The router sends an ARP-query in order to resolve an IP address. It then receives a single or multiple ARP-replies and keeps the first one received. If the chosen destination goes down, the corresponding ARP-entry in the router will ultimately timeout. Until it times out, the destination will still be considered as reachable, but all packets sent to it will be dropped. In the case of anycast servers, the desired behavior is of course to switch promptly from one server to another when the former goes down. The only way to achieve this is to use small timers [26].

We see that IPv4 ARP and IPv6 ND differ only in how promptly they can discover that a destination has become unreachable. ND uses a dedicated mechanism sending probe messages for that purpose, whereas ARP simply uses timers. Besides the fact that ARP will not detect a failure as quickly, timeouts of ARP entries for anycast servers might kill stateful connections. Every time there is an ARP timeout a new ARP request will be sent out and if a reply is received from a different anycast server then subsequent packets destined to the anycast server will go to that server. This will kill any stateful connection that has been previously setup to the anycast server.

Consider a scenario where multiple hosts sharing a single anycast address are connected to the same network segment. The router on this segment is responsible for distributing packets destined to this anycast address. One approach is to enhance the ARP or ND mechanisms at the router to maintain all link-layer addresses that map to the shared address. One of the policies described in Section 3.2 can then be used to forward the packets. Only the router to which the hosts sharing the anycast address are connected needs to implement these changes in ARP or ND.

An alternative to modifying link-layer mechanisms for the purposes of load distribution, is to use routing mechanisms either through configuration at the router or by allowing the hosts to participate in the routing protocols. We describe these two alternatives in the context of Figure 2. Assume that U_1, U_2 and U_3 are the 3 unique unicast IP addresses that are bound to each of the hosts with anycast address A as shown in Figure 2. Then we can do load-distribution among these 3 servers in one of two possible ways: (1) set-up multiple static routes in router E with destination A and next-hops U_1, U_2 and U_3, or (2) let nodes U_1, U_2 and U_3 run a routing protocol and advertise reachability to A (through external route advertisements in the case of OSPF, or regular updates for RIP). In both cases, if router E supports multipath routing, it will perform load-distribution among multiple equal-cost routes to A.

In this section we outlined a few mechanisms by which the router could forward anycast datagrams. However, regardless of the mechanism used, it is always possible for the routing tables or the link-layer address mappings to change. This will result in packets from the same flow being sent to different destinations. In the next section we propose modifications to the protocol processing stack in the host to ensure that anycast packets belonging to a single flow are always routed to the same end host.

4 Maintaining Stateful Connections

From the previous discussion it is evident that the anycasting service does not require routers to maintain any state. In other words the network has no obligation to send two successive datagrams destined to the same anycast address to the same end station. This may be a significant problem for applications that maintain stateful connections to remote hosts using anycast addresses. To alleviate this problem we need to make changes either in the host protocol stack or in the routers so that anycast packets belonging to

the same flow are treated in a consistent manner. In this section we discuss modifications to protocol processing at the host for this purpose.

In [26] Partridge et al. suggest a modification to TCP that would allow all packets belonging to a TCP connection to be delivered to the same host even when the destination address is an anycast address. In their solution, when a host initiates a TCP connection to a remote anycast address, it treats the destination address as a wildcard address in its local protocol control block, but uses the anycast address as the destination address in the first packet of the TCP connection (SYN packet). When the SYN packet is received by one of the hosts in the anycast group, it responds by sending a SYN-ACK where it uses one of its unicast addresses as the source address. Upon receiving the SYN-ACK, the initiating host instantiates the remote address of the connection in its local protocol control block with the unicast address of the remote host and uses this address as the destination address in subsequent exchanges. This ensures that after receiving the SYN-ACK the TCP connection is bound to a single remote host.

The problem with the scheme described above is that it requires changes to the TCP semantics. Protocol stacks at the initiator of the connection, as well as the recipient, have to be modified in order to realize such changes in the TCP semantics. In a client/server model, both client (initiator) and server (recipient) must be changed, which is not desirable in an infrastructure such as the world-wide web, where the number of clients is much larger than the number of servers.

Another approach suggested in [29], in the context of multi-homing can also be adapted to bind an anycast address to a unicast address. The idea here is to use a new IP option, named Source Identification Option, to inform clients that a particular communication initiated through the use of an anycast address should proceed with the use of the unicast address of one of the anycast group members. In this scheme, the initiator of the connection sends the datagram to the anycast address that is ultimately received by one of the group members. In its reply, the remote host puts one of its unicast addresses (the remote host may be multi-homed) in the source identification option. Upon receipt of the reply, the initiator can replace the anycast destination address with the unicast address supplied by the remote host. The drawback of this solution is that it requires a new IP option to be defined.

We propose an alternative approach to maintain the association between an anycast address and the unicast address of one of the hosts in the anycast group for the duration of a connection. The advantage of this approach is that it uses an existing IP option, namely Source Route Option, available in both IPv4 and IPv6 and requires minor modifications to protocol processing. Furthermore, this modification impacts only the recipient of a TCP connection, making it suitable to a large client/server environment such as the world-wide web, where only the servers need to be changed.

In the following we describe the proposed solution and its implementation in AIX 4.2 TCP/IP stack.

4.1 Modification to Transmit Path

Let us first consider the case where TCP is used as the transport protocol. Consider a host with a unicast address C initiating a TCP connection to a remote address A, that happens to be an anycast address. The initiating host first sends a SYN packet with A as the destination address. The network routes the packet to one of the hosts in the anycast group. The TCP module of the remote host processes the SYN packet and generates a SYN-ACK with S as the destination address and A as the source address

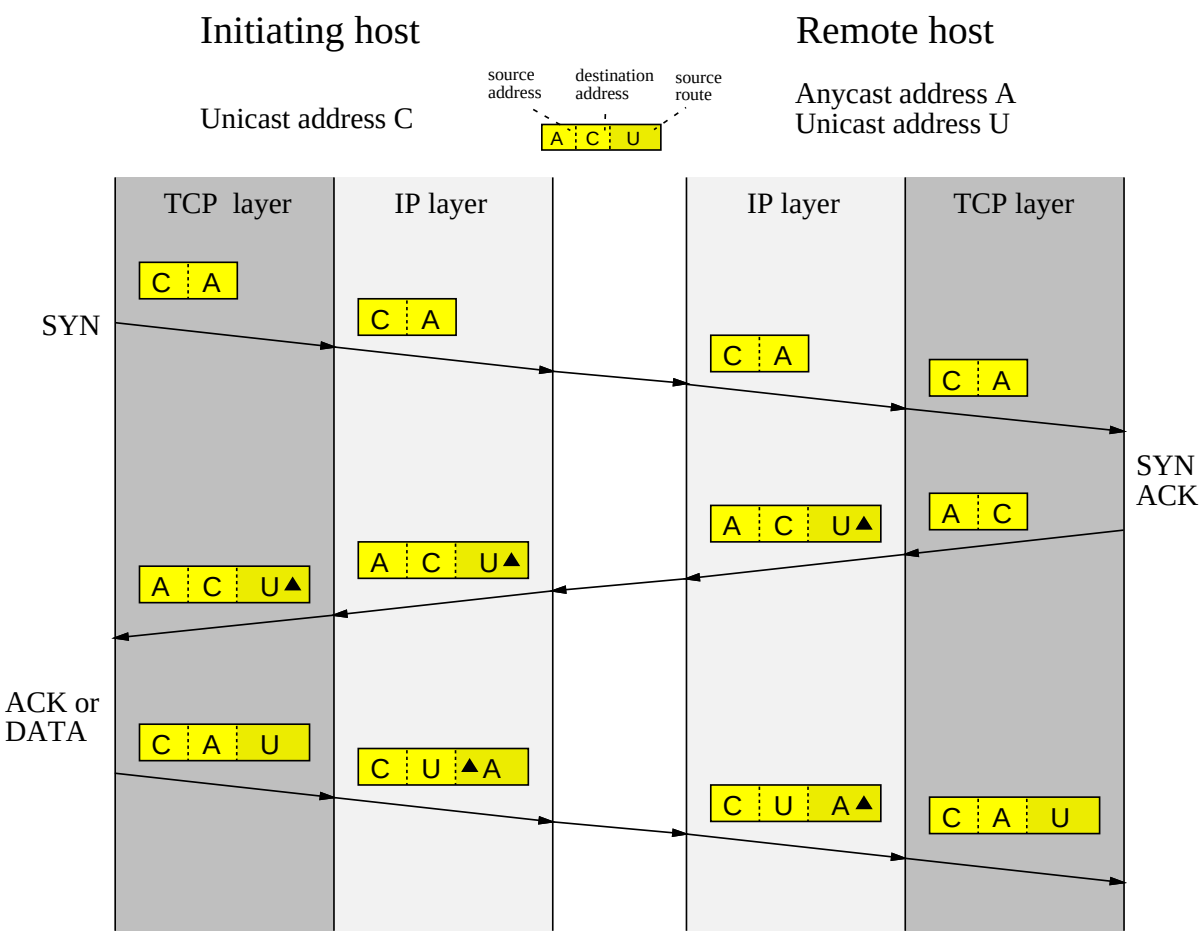


Figure 3: Source route insertion in the transmit path.

and passes that on for processing by the IP layer. The IP layer determines the appropriate interface on which the packet should be forwarded. It also recognizes that the source address of the packet is an anycast address. At this point, it inserts a source route indicating that the packet has been forwarded through one of the unicast addresses of the outgoing interface, say U. Note that the check to determine if an address is an anycast address and insertion of the source route is a deviation from standard IP layer processing at the host. When the SYN-ACK is received at the other end, it appears to have arrived from A via U. The IP layer processes the source route and passes that on to the TCP layer, which stores it in the TCP control block. In subsequent exchanges to the remote host, the source route is reversed and used to route the packets. This causes the packets to be routed through the unicast address of the remote host and establishes a binding between the anycast address A and the unicast address U for the lifetime of the connection.

Figure 3 explains the scheme described above in more details. It shows the state variables, namely the source and destination addresses, and the source route at different layers of the protocol stacks at both ends of the connection. Note that while the state variables are part of the TCP state for the life time of the connection, at the IP layer they are part of the state associated with the current packet being processed.

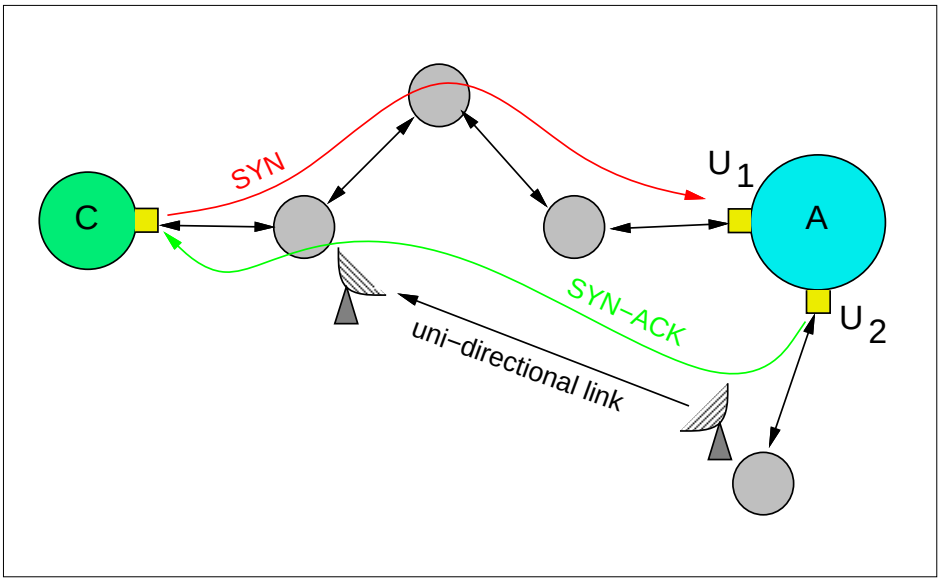


Figure 4: Asymmetric Routing and multi-homed host.

As shown in Figure 3, the source route is inserted by the IP layer at the remote host while processing the SYN-ACK packet. Note that the source route consists of only one hop, the unicast address of the interface. Hence, IP option processing has to be performed only at the end-host, not at the routers on the path. The filled triangle on the right reflects the position of the next hop pointer in the source route option field in the IP header and indicates that this hop has already been traversed. When the SYN-ACK reaches the destination, the IP layer processes the source route and passes it on to the TCP layer where it becomes a part of the TCP connection state. In the packet that follows, the TCP layer reverses the source route (it is a list consisting of a single hop in this case) and passes that on to the IP layer along with the source and destination addresses. The IP layer determines that the next hop for the packet is U and puts that as the destination address. It also puts A, the ultimate destination, in source route option field in the IP header. Note, that the filled triangle is on the left of A. This indicates that the next hop to be traversed is A. When this packet is received at the remote host, as a part of standard source route processing it should swap A and U and place the filled triangle past U indicating that this hop has been traversed. It should then forward the packet to the final destination, which happens to be the same host. In Figure 3 we do not show the last step. It is in fact an optimization that saves us from a second round of processing at the IP layer. The IP stack at the remote host has been modified to recognize that the ultimate destination is itself. Consequently, it processes the packet and passes the source route to the TCP layer which stores it in the connection state. In subsequent exchanges, the TCP layer at the remote host reverses the source route and passes that on to the IP layer to be put into the IP header.

4.2 Modification to Receive Path

Although the solution described above works in most cases, it may lead to problems when the remote host is multi-homed and routing is asymmetric. To understand this problem, let us consider the scenario

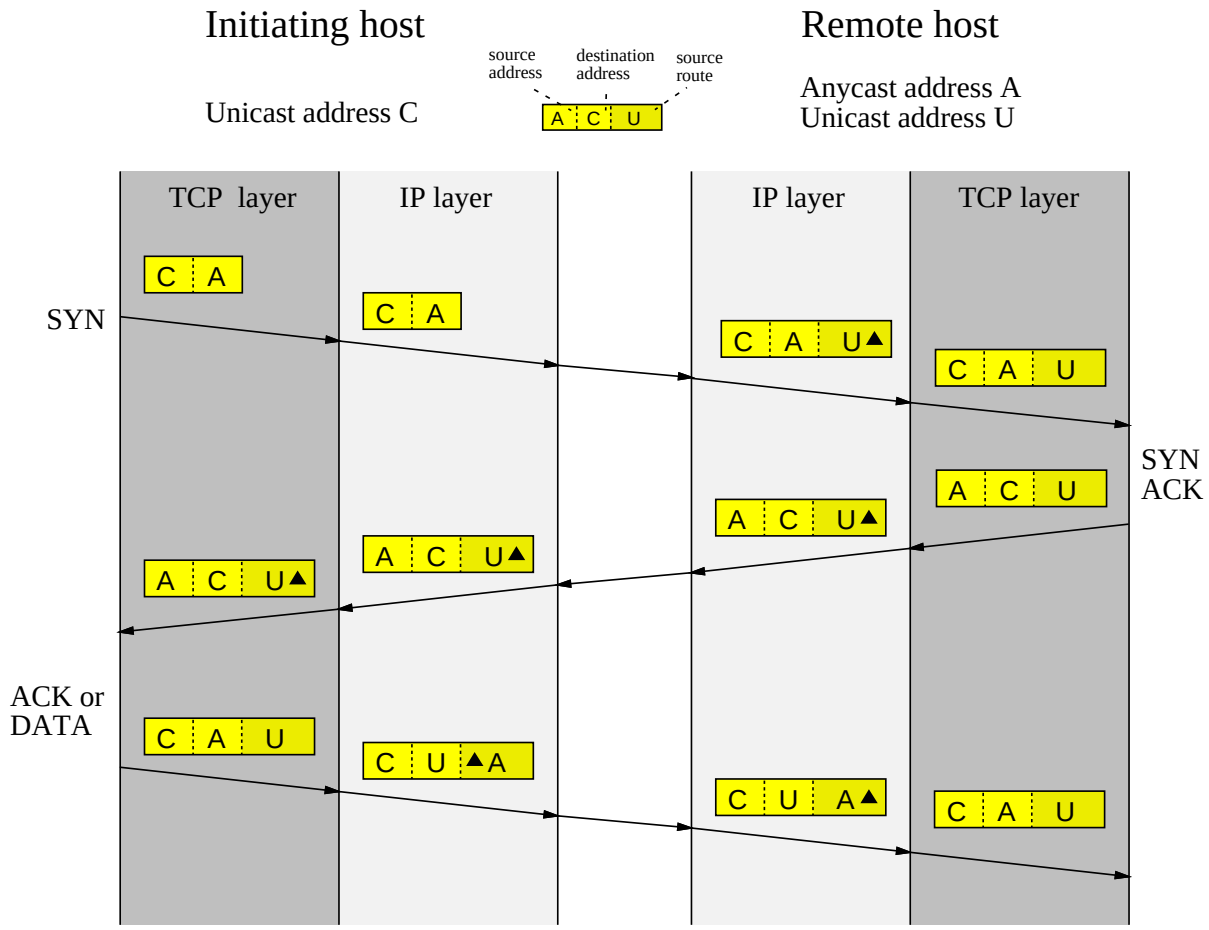


Figure 5: Source route insertion in the receive path.

pictured in Figure 4 where the remote host has two interfaces with unicast addresses U_1 and U_2. In addition it also has an anycast address A. We assume that routing is asymmetric and packets from C to A are routed via interface U_1 while packets from A to C are routed via U_2. Now let us trace through the connection setup phase between C and A. As in the above description, the connection is initiated by a SYN packet with source address C and destination address A. In response, the TCP module at A generates a SYN-ACK with A as the source address and C as the destination address. The IP layer at the remote host traps the packet and recognizes that A is an anycast address and attempts to add a source route to indicate that the packet has been routed through an unicast address. The dilemma now is which one of U_1 and U_2 to use as the unicast address. The obvious choice is the interface which routing chooses as the outgoing interface for the packet destined to C. In our example U_2 is the outgoing interface for destination C. If U_2 is used as the intermediate hop, when C reverses the source route for subsequent exchanges with A, the packets are routed through U_2 instead of U_1. However, U_1 happens to be the route of choice for packets generated by C and destined to A. If routing is asymmetric and U_2 is unreachable from C, the connection may be terminated.

An easy fix for this problem, is to insert the source route when a packet with A as the destination address is received at the remote host. As a part of IP layer processing, the remote host can insert a source route with the incoming interface U_1 as the intermediate hop and pass that on to the upper layer. The upper layer then can store the source route in the connection control block and use the reversed route in subsequent exchanges. Care has to be taken to make sure that the source route is inserted only when appropriate. That is, if the packet already contains a source route with one of the unicast addresses of the host as the penultimate hop, no changes are required. For packets already carrying a source route option but where the penultimate hop is not one of the unicast addresses of the host, it has to be properly inserted in the source route.

Figure 5 shows the processing involved in the TCP and IP layers of the source and destination in more details. We use the same notation as in Figure 3. In this case the SYN packet is trapped at the IP layer in the remote host. It recognizes that the destination address in the packet is one of its anycast addresses and inserts a source route with a single hop U. Note that the filled triangle is placed past U indicating that this hop has already been traversed. This source route is passed on to the TCP layer which stores it as part of the connection state. When the TCP layer at the remote host generates the SYN-ACK in response to the SYN packet, it passes the reversed source route to the IP layer along with the source and destination addresses. The IP layer adds the source route in the header and forwards the packet. The source route processing at the IP layer presented here is a deviation from the source route processing in a standard IP stack and represents an optimization used in our implementation. In a standard implementation, the packet will pass through the IP layer twice, first time with U as the destination address and C as the next hop, and the next time with C as the destination address. In our implementation, these two passes are integrated into a single pass through the IP layer. When the SYN-ACK is received at the destination, the IP layer processes the source route and passes that on to TCP. The TCP layer saves the source route in its connection state and use the reversed route in subsequent exchanges.

4.3 Stateful Communication over UDP

In the discussion above we assume that TCP is the transport protocol. Although UDP does not have the notion of a connection, applications often use UDP as the underlying transport protocol for stateful communication. An application using UDP as the transport protocol has two modes of operation. It can either bind a transport layer end-point (*socket*) to a specific remote destination address and send data to only that address, or use the same end-point to send data to multiple remote addresses. Similarly, an application can use the same socket to receive data from multiple sources. Appropriate handling of one-to-many and many-to-one semantics of UDP sockets requires an enhancement to the source routing solution described above.

If a host uses a single UDP socket to communicate with multiple anycast hosts, it may have to store multiple source routes, one for each anycast destination, as part of its protocol control block. If the anycast host is not multi-homed and inserts source routes in the transmit path, it need not store any source route as a part of its connection control block. If however, the anycast host is multi-homed, it is advisable that it inserts the source route in the receive path and store that as a part of the protocol control block. In that case, it may need to store multiple source routes with an unconnected socket.

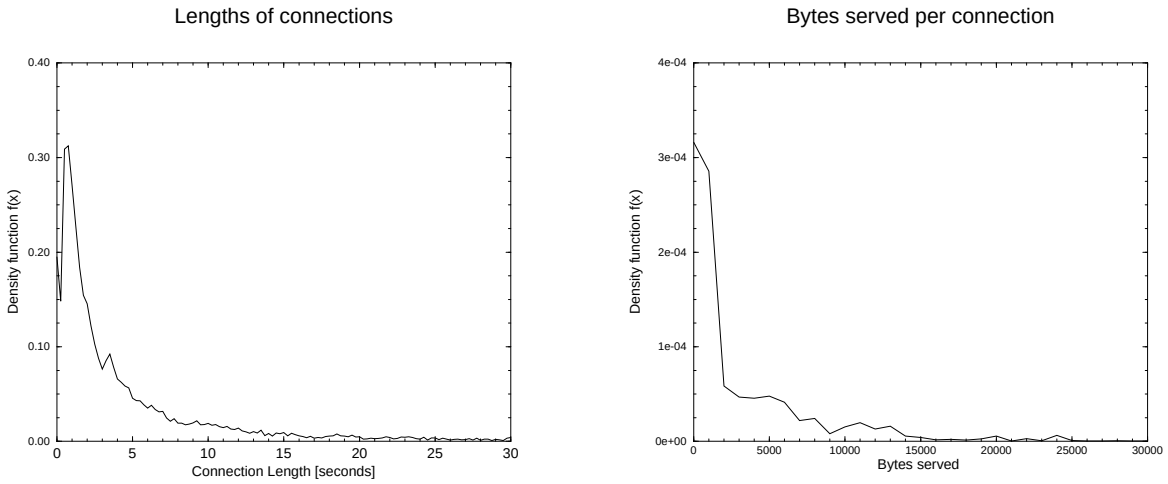
When multiple source routes are associated with a single socket, the appropriate source route has to be used when a packet is sent out. For the sake of simplicity, it may be a good idea for the clients to use a connected socket while communicating with the server over a stateful connection. However a server has to use a non-connected UDP socket in order to serve multiple clients. A possible optimization on the server side is to store the source route (unicast address of the incoming interface) associated with the last packet received. While sending a packet out, it can check if the current source route can be used to route the packet to the destination. If that is the case, it uses the cached source route. Otherwise, it picks one of its unicast addresses, creates a source route, and inserts it in the packet.

We have implemented the changes described in this section in the AIX 4.2 TCP/IP protocol stack. We have added a feature that allows one to configure an address as an anycast address. This is in essence an enhancement to IP aliasing feature supported by most TCP/IP stacks. Using the IP aliasing feature, one can specify one or more aliases for an interface. However, an anycast address is more than a simple alias. A distinction between the aliased unicast address and an anycast address is required since packets with an anycast address as a source or destination address are trapped by the host identified with that address and processed differently. As explained above, changes are required only at the servers, that is, hosts with anycast addresses. On the receive side the changes are contained within the `ipintr` function and on the transmit side changes are restricted to the `ip_output` function.

4.4 Pragmatics of Using Source Routing

Source routing offers flexible means to provide useful services in the Internet. There are, however, security and performance implications of using source routing. In the absence of other security measures, source routing makes it easier to spoof the source of an IP packet. A malicious user can divert the packets that are destined to a particular host by inserting a source route through itself. However, source routing is not the only way to divert packets to a malicious host; the propagation of incorrect routing information or the use of ICMP redirect messages are examples of other possible attacks [3]. In fact it is well understood that reliance on the source address in a packet for any form of authentication can be extremely dangerous [3]. Currently, the issue of security in the Internet is being widely discussed in the scientific community and there are several standardization efforts in place at the IETF. End-to-End security can be provided both at the IP layer [1] as well as the transport layer [13]. In conjunction with any of these security mechanisms source routed packets provide the same level of security as regular IP packets.

Another potential problem with source routing is the overhead of processing IP options at the intermediate routers. This has been a problem with IPv4 since options can be inserted in a packet in any possible order. Hence, even if none of the options present in a packet are relevant to an intermediate router, it cannot determine this without parsing the entire list of options. To improve performance some routers treat packets with options with a lower priority than packets without options. This problem however has been corrected in IPv6 [11]. IPv6 defines a specific extension header for source routing. An intermediate router is not required to look at this extension header unless the destination address in the packet matches one of its interfaces. The solution proposed in Section 4 inserts a source route that only includes the end host. Thus in IPv6 only the end host is required to examine the routing header and the intermediate routers are not impacted in any way.



(a) Connection life time.

(b) Number of bytes transferred.

Figure 6: Distribution of connection life time and number of bytes transferred.

5 Simulation Results

In the previous sections, we discussed possible enhancements to routing and host protocol processing that allow widespread use of network layer anycast without modification to the existing applications and their communication semantics. In this section, we focus on applications of anycast to provide improved network services. We consider world-wide web (WWW) as an example, and show how anycast can be used as a mechanism for load distribution and service location.

Exponential increase in traffic volume at popular web sites has forced site administrators and network service providers to use innovative means to improve the scalability of the web infrastructure. A commonly used technique to improve scalability is server replication. For popular web sites, replicas are placed at different strategic locations to exploit user locality. Each replica often consists of a cluster of server nodes which also helps improve the scalability and reliability of the server. Server replication however, leads to new problems. For the users, the problem is to find the best mirror site. For the network, the problem is to distribute the load among different replicas. A number of schemes have been suggested for server location and load distribution. In the rest of the section we discuss how network layer anycast can evolve to be a more scalable and potentially better solution to this problem.

One of the commonly used techniques for server location and coarse grain load distribution is to use enhanced versions of Domain Name Service. A Domain Name Server (DNS) can resolve the same name to different IP addresses for the purpose of load distribution. Round Robin DNS and application layer anycasting [4] are examples of this scheme. The problem with these schemes is that intermediate name servers cache the resolved name-to-IP address mapping. If these mappings are cached for a long time, fine grain load distribution becomes difficult. If the caches are timed out quickly, the load on the DNS and network increases significantly. The other major problem with DNS based schemes is that they fail to factor in network and server load dynamics in making load balancing decisions. The Host Proximity

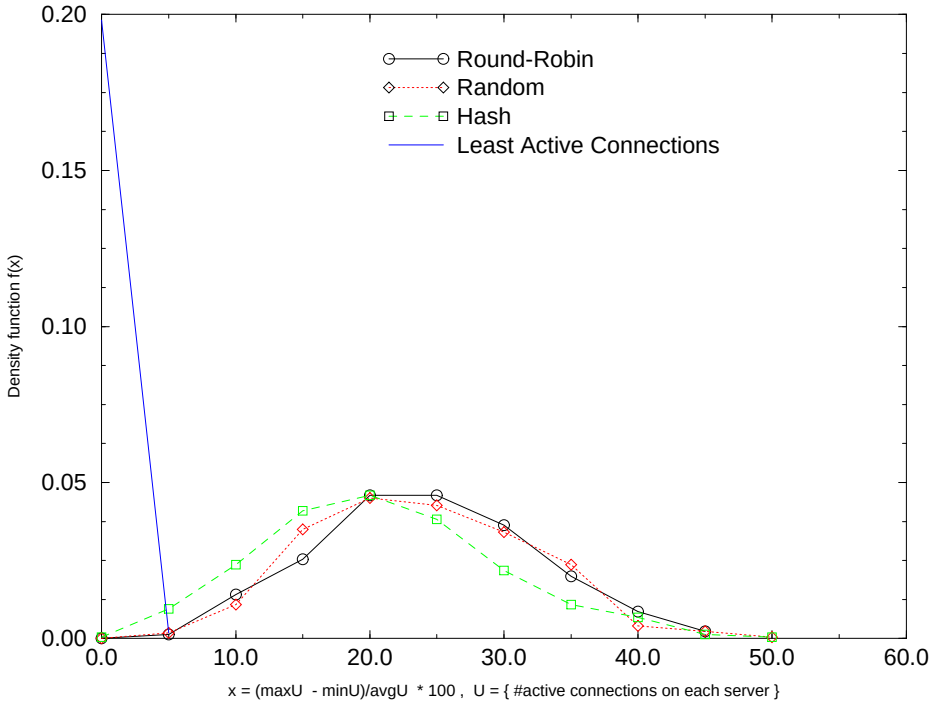


Figure 7: Distribution of number of active connections

Service (HOPS) and Distributed Director alleviate a part of this problem by participating in routing protocols and by using other mechanisms to monitor server and network loads [8, 14]. However, for fine grain load balancing, they require the clients to consult the HOPS server (or the distributed director), each time a new connection is initiated. This essentially adds an extra round trip time to the HOPS server to the connection setup time. Given that most of the connections to web servers are relatively short, this overhead may significantly reduce the advantages of using this service.

For fine grain load balancing in a server cluster, many popular web sites use a connection routing front-end that distributes connections to different server nodes. It maintains a view of the load on each server in order to forward new connections to the most appropriate server. One can combine a DNS based scheme with connection routing [12] front-ends in order to get the best of both approaches. However, this does not solve the problem of locating the closest server for the requesting client. Additionally, since all connections to a server cluster have to pass through the connection router, it is a single point of failure and may become a bottleneck at high loads.

We argue that network layer anycast provides a more scalable solution to the server location and load distribution problem as compared to the schemes described above. In an anycast based scheme, all mirror sites of a server share a single anycast address. When a client sends a connection request to the anycast address of the server, it automatically gets routed to the nearest mirror site. Unlike HOPS and DNS based

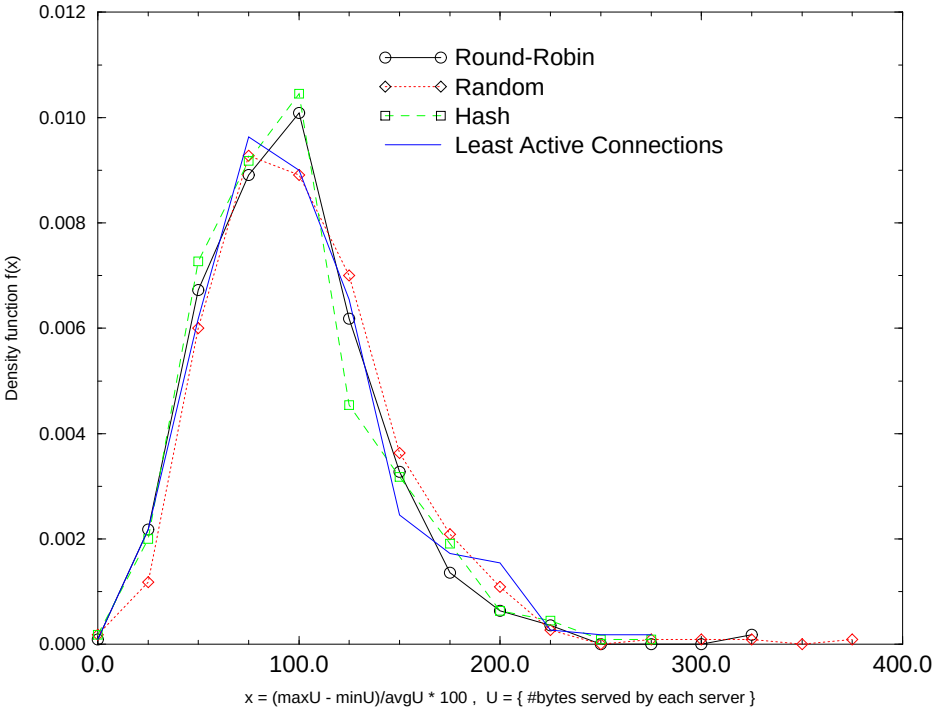


Figure 8: Distribution of number of bytes transferred.

schemes, network layer anycast service is a part of the routing infrastructure. It requires no additional mechanisms to determine the mirror site closest to a requesting client. It not only helps locating the closest server, it also distributes load among the replicas without introducing new servers (e.g. HOPS server, Distributed Director). It does not add any extra load on the networks in terms of DNS or HOPS queries. The clients also save a round trip time to the HOPS server or DNS. Furthermore, it is quite efficient in distributing load with very fine granularity. In fact, with a little sophistication, network layer anycast can be as effective as a connection router for fine grain load distribution without the overhead of per-connection state maintenance. In the following we present some simulation results in support of this claim.

To demonstrate fine grain load distribution using anycast, we use an experimental setup where a router is used to distribute load among N servers connected directly to it. In the specific experiment reported in this paper, 5 server nodes sharing a single anycast address are connected to the router. We assume that the servers implement the modified protocol stack which pins a connection to a server node. Note that as far as the router is concerned, only the first packet of a connection (TCP SYN) has the anycast address as the destination address. The router treats the route leading up to the anycast address as a multi-path route consisting of five different routes. We consider 3 different dispatching policies: (1) round-robin, (2) random, and (3) hash-based that the router can use to distribute load among the server nodes. In the round-robin

scheme, packets destined to anycast address are forwarded to one of the routes in a round-robin manner. In the second scheme, the packets are forwarded to one of the randomly chosen routes. In the hash-based scheme, the route is chosen by hashing on the source address and the port number. Since in the hashing scheme packets coming from a specific port of a certain host are always routed to the same server node, pinning of connections by the server is unnecessary. Note that in all the three cases, the router need not maintain any connection state.

We use the *tcpdump* traces collected from the Official 1996 Olympic Web Server [2] as the traffic profile for this experiment. Traces were collected for approximately 2 weeks in 20-minute segments. The entire trace consists of approximately 60 million HTTP requests, from about 725,000 clients. In figure 6 we plot the distribution of connection holding time and amount of data transferred for a 4-minute segment from the trace. As seen in the figure, most of the connections are short-lived and transfer small amounts of data. This observation is representative of similar observations from other segments of the trace and is consistent with traces collected at other web sites.

Figures 7 and 8 show the relative performance of different policies in distributing load. In this experiment, connections arriving at the server complex are dispatched by the router to one of the server nodes using one of the four policies – (1) round-robin, (2) random, (3) hash-based, and (4) node with least active connections. Anyone of first three policies can be used in anycast-based scheme for load distribution. The fourth policy is used by many connection routers as a heuristic approximating the ideal load distribution policy.

The simulation keeps track of the number of active connections as well as the total bytes transferred at each of the 5 servers. The number of bytes transferred in half second intervals as well as the number of active connections are sampled every half second. Figure 7 plots the probability density function of load imbalance in terms of number of active connections. In this figure, the X-axis is the difference between the maximum and minimum number of active connections at each sampling point, normalized to the average number of connections active at that time. The Y-axis is the probability density of maximum load imbalance among different nodes. Clearly, the closer the density function is to the Y-axis the better is the load balancing. As expected, the least active connection policy performs the best in this case. It manages to keep the load imbalance within 5% of the average load in the worst case. The performance of other policies are similar to each other. The load imbalance never exceeds 50% of the average load and is below 30% in most cases.

Although the number of active connections is a good measure of server load, the amount of data transferred is a more appropriate metric for network load. In Figure 8 we compare the four policies in terms of volume of data transfer. In this figure, the X-axis is the difference between the maximum and minimum number of bytes transferred in the last 0.5 second, normalized to the average number of bytes transferred by a node during the same time. Here, we observe that all four policies show very similar behavior. In this case, the maximum load imbalance is within 125% of the average load in most instances. However, there are points in time when load imbalance shoots over 300% of the average load. Fortunately, such cases are very rare. From these results, we conclude that in terms of network load distribution, anycast-based schemes perform as well as load distribution schemes that maintain information about connection states.

6 Conclusion

In this paper, we examined different aspects of using IP anycast as a mechanism for load distribution and service location in the Internet. We proposed a scheme that allows applications to transparently use anycast services without any change in their communication semantics. The proposed scheme does not require any modification to the routers and routing protocols. However, to exploit the full potential of anycast, certain enhancements to multi-path forwarding and/or address resolution are desirable. We described these enhancements along with some recommendations on how advertisements for anycast addresses can be distributed using the existing routing protocols. On the host side we proposed modifications to the protocol processing to ensure that packets belonging to a single flow are consistently routed to the same end-host. These modifications are limited to changes at the IP layer of the recipient of the TCP connection, making this scheme suitable to a client/server environment. We have implemented the proposed solution in the AIX 4.2 TCP/IP stack. Using a trace based simulation, we also evaluated several anycast-based load-distribution policies. Based on this study, it is fair to conclude that IP anycast should be considered seriously as a candidate mechanism for load distribution and service location in the Internet.

References

[1] R. Atkinson. RFC 1825: Security architecture for the Internet Protocol, August 1995.

[2] H. Balakrishnan, M. Stemm, S. Seshan, and R. H. Katz. Analyzing stability in wide-area network performance. In *SIGMETRICS'97*, Seattle, June 1997.

[3] S. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communication Review*, April 1989.

[4] S. Bhattacharjee, M. H. Ammar, E. W. Zegura, V. Shah, and Z. Fei. Application-layer anycasting. In *Proceedings of the IEEE INFOCOM '97*, 1997.

[5] K. Birman and T. Joseph. Reliable communication in the presence of failures. *ACM Transactions on Computer Systems*, 5:47–76, February 1987.

[6] T. Brisco. RFC 1794: DNS support for load balancing, April 1995.

[7] B. Chinoy. Dynamics of internet routing information. In *Proceedings of SIGCOMM'93*, pages 45–52, September 1993.

[8] Cisco distributed director. White Paper, 1996. Cisco Systems.

[9] P. Danzig, D. Delucia, and K. Obracza. An analysis of wide-area name server traffic. In *Proceedings of ACM SIGCOMM'92*, Baltimore, MD, August 1992.

[10] P. Danzig, D. Delucia, and K. Obracza. Massively replicating services in wide-area internetworks. Technical report, University of Southern California, 1994.

[11] S. Deering and R. Hinden. RFC 1883: Internet protocol, version 6 (IPv6) specification, January 1996.

[12] D. Dias, W. Kish, R. Muhkerjee, and R. Tewari. A scalable and highly available web server. In *Proceedings of the IEEE Computer Conference (COMPCON)*, Santa Clara, March 1996.

[13] T. Dierks, P. L. Karlton, A. O. Freier, and C. Kocher. The TLS protocol, version 1.0. `draft-ietf-tls-protocol-03.txt`, May 1997. Work in progress.

[14] P. Francis. A call for an internet-wide host proximity service (hops). URL: `http://www.ingrid.org/hops/wp.html`, 1996.

[15] R. Guérin, S. Kamat, A. Orda, T. Przygienda, and D. Williams. QoS routing mechanisms and OSPF extensions. `draft-guerin-qos-routing-ospf-01.txt`, March 1997. Work in progress.

[16] J. Guyton and M. Schwartz. Locating nearby copies of replicated internet servers. In *Proceedings of ACM SIGCOMM'95*, pages 288–298, 1995.

[17] C. Hedrick. RFC 1058: Routing Information Protocol, June 1988.

[18] C. Hedrick. An introduction to IGRP. Cisco Technical Report #1, August 1991.

[19] C. Huitema. *Routing in the Internet.* Prentice-Hall, 1995.

[20] J. L. R. Ford and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, NJ, 1962.

[21] M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, and L. Jones. RFC 1928: SOCKS protocol version 5, April 1996.

[22] G. Malkin. RFC 1388: RIP version 2 carrying additional information, January 1993.

[23] P. Mockapetris. RFC 1035: Domain names — implementation and specification, November 1987.

[24] J. Moy. RFC 1583: OSPF version 2, March 1994.

[25] T. Narten, E. Nordmark, and W. Simpson. RFC 1970: Neighbor discovery for IP Version 6 (IPv6), August 1996.

[26] C. Partridge, T. Mendez, and W. Milliken. RFC 1546: Host anycasting service, November 1993.

[27] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics.* PhD thesis, Computer Science Division, University of California, Berkeley, April 1997.

[28] D. Plummer. RFC 826: Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit ethernet address for transmission on Ethernet hardware, November 1982.

[29] M. Shand and M. Thomas. Multi-homed host support in IPv6. Internet Draft, June 1997.

[30] J. Veizades, E. Guttman, C. Perkins, and S. Kaplan. RFC 2165: Service Location Protocol, June 1997.