Versatile Anycasting with Mobile IPv6

Michal Szymaniak Guillaume Pierre Maarten van Steen

Department of Computer Science, Vrije Universiteit Amsterdam
De Boelelaan 1081a, 1081HV Amsterdam, The Netherlands
Email: {*michal,gpierre,steen*}@*cs.vu.nl*

Abstract

Anycasting was introduced to facilitate efficient communication between distributed Internet services and their clients, as it allows client requests to be automatically routed to nearby service instances. However, even though several anycast implementations have been proposed, their various limitations prevent them from being widely adopted by large-scale distributed systems.

This paper identifies the key limitations of existing anycast implementations, and proposes how to implement anycast such that all these limitations are addressed without harming the performance of anycast communication. Our solution relies on address-translation capabilities present in modern operating systems. These capabilities have originally been designed for communication with mobile nodes. However, we demonstrate that one can exploit them to implement versatile anycasting at low cost.

1 Introduction

Anycast is a network addressing and routing scheme whereby data are routed to one of many possible nodes forming an anycast group [24]. The chosen node is typically the "nearest" or "best" to the data sender as viewed by the network topology, which enables Internet services to easily redirect each of their clients to its proximal servicing facility.

Anycasting by nature ensures communication locality, which makes it very attractive to globally distributed systems. Various research efforts have proposed exploiting anycast for content delivery [14], balancing load across root DNS servers [17], or implementing multicast rendez-vous points [21]. When implemented properly, anycasting can also be used to develop distributed virtual servers in which multiple nodes share the same contact address, or to identify nearby members of peer-to-peer overlays.

However, even though several anycast implementations have been proposed, none of them have been widely adopted in large-scale distributed systems, which still tend to implement anycast-like functionality on their own [4, 6]. We believe that the problem lies in a number of limitations exhibited by existing and proposed anycast implementations. For example, many implementations do not provide fine-grain control over traffic switching, cannot handle failures within anycast groups, and do not support connection-oriented communication properly. While all these limitations can be addressed using a proxying frontend, following this approach makes anycasting inefficient and leads to problems with scalability.

In this paper, we demonstrate that anycasting can be implemented such that all these limitations are addressed, yet without affecting the performance of anycast communication. Our solution exploits address-translation mechanisms provided by Mobile IPv6, which has originally been proposed to enable communication with mobile nodes. We show that these mechanisms can also be used to switch traffic among nodes forming anycast groups in a controlled manner. Such switching enables our solution to transparently hand off traffic targeting the address of a given anycast group to the address of any node within that group, which effectively provides anycast functionality.

Our anycast implementation preserves the good properties of existing implementations. First, it exploits standard network protocols, and so is readily available for global-scale deployment. Second, it enables direct communication between nodes forming anycast groups and their clients, which makes that communication very efficient.

Equally important, our solution also addresses the limitations exhibited by the current implementations. First, it provides fine-grain control over traffic switching. Second, it tolerates rapid changes in the composition of an anycast group, as clients serviced by a node leaving the group can transparently and swiftly be taken over by any other node within the group. Third, it consistently routes traffic between an anycast group and its clients, thus allowing individual nodes within that group to maintain connection state. In our solution, switching a client to another node within the group causes that connection state to migrate as well, implying that the client connection remains intact. The measurements performed on our prototype anycast testbed demonstrate that the only overhead of our implementation is the short delay caused by traffic switching, which is a linear function of network latencies between the nodes involved.

The possibility of implementing anycast functionality using Mobile IPv6 has already been identified in two earlier publications. The first one proposes to exploit mobile extensions of IPv6 to route requests in content delivery networks [3], whereas the second one sketches how to redirect clients to anycast nodes using Mobile IPv6 signaling [16]. However, both these studies build on early versions of the Mobile IPv6 specification, which differ significantly from the final protocol covered by this paper. Also, besides employing Mobile IPv6 to implement (relatively straightforward) one-time traffic switching, we also demonstrate how Mobile IPv6 can be exploited to ensure anycast address stability and to implement multi-layer wide-area client handoffs. Finally, while the two earlier studies are purely theoretical in nature, we base our considerations on practical experience with our prototype anycast testbed.

For lack of space, the detailed description and evaluation of our proposed techniques could not be included in this paper. We refer the interested reader to the accompanying technical report [31]. Instead, this paper focuses on a high-level description of our approach and on extensive analysis of its properties in comparison with existing approaches.

The rest of this paper is structured as follows. Section 2 describes the requirements that must be met by any practical anycast implementation. Section 3 discusses to what extent these requirements are met by a number of existing solutions. Section 4 presents our anycast implementation, and explains why it meets the requirements better than its earlier counterparts. Finally, Section 5 concludes.

2 Requirements

Anycast is a network addressing and routing scheme whereby data are routed to one of many nodes forming an anycast group [24]. The chosen node is typically the "nearest" or "best" to the data sender as viewed by the network topology. For the sake of brevity, we refer to members of an anycast group as to "anycast nodes."

Every anycast implementation must meet two functional requirements. First, one must organize a number of nodes into an anycast group. Typically, the group provides some service. Whereas the service details are irrelevant, the implementation must allow the anycast nodes to be distributed over a wide-area network, as anycast is typically used in Internet-scale deployments [14, 17, 21].

The second functional requirement is that each anycast group can be assigned a contact handle, such as an IP address or DNS name, which can be used by the clients to send traffic to that group. The anycast implementation must ensure that traffic sent by a given client to a contact handle reaches exactly one anycast node within the respective anycast group, preferably the one closest to the client in terms of some network distance metric. Selecting such a node and re-routing the traffic should remain transparent to the client even when the composition of the anycast group changes.

In addition to meeting the above functional requirements, an anycast implementation must also have a number of non-functional properties to be useful in practice. First, the communication between the clients and the anycast nodes must be efficient in the sense that anycasting cannot introduce too much overhead in comparison to direct communication between a client and an anycast node. Second, the anycast implementation should be scalable enough to handle global communication. Finally, the anycast deployment should not require any changes to the existing Internet infrastructure.

All these base requirements are met by the standard routing-based anycast implementation. However, we observe that it is relatively seldom used by Internet applications, which tend to implement anycast-like functions on their own. This is, for example, what happens in content delivery networks, whose request routing subsystems work very similar to anycast. We believe that anycast implementations meeting only the above requirements are still not flexible enough to be widely adopted by contemporary distributed systems, which typically expect anycast to provide much more than primitive traffic scattering [33].

The primary additional function that many distributed applications require from anycast is fine-grain control over dispatching traffic to individual nodes. A practical anycast implementation should therefore enable an anycast group to route anycast traffic according to any metric and not only to network distance between clients and anycast nodes. For example, in classical load-balancing schemes, traffic is routed based on both network distance and the current load of each node servicing the traffic [9, 25].

Another useful property of a practical anycast implementation is its resilience to potentially frequent changes in the composition of anycast groups, as large-scale distributed systems are increasingly often composed of unreliable nodes [23]. In particular, an anycast group should be able to quickly adapt to ungraceful departures of anycast nodes, such that the group traffic is always serviced by the nodes remaining in the group. This requires that the client-to-node traffic-control mappings are not only fine-grain, but also that they can be rapidly updated.

Since anycast effectively converts a group of anycast nodes into a single virtual node, it is also desirable to make the communication with anycast groups as reliable as with regular nodes. This means in particular that updating the traffic-control mappings inside an anycast group should not break the communication between that group and its clients. However, when clients communicate with anycast groups using connection-based protocols such as TCP, rapid traffic switching between anycast nodes might result in such connections to be accidentally terminated [29]. A practical anycast implementation should prevent such problems by enabling the anycast nodes to transparently handoff client connections between each other so that communication with the anycast group is not disrupted upon traffic switching.

Finally, deploying anycast in the current Internet should be simple, and should not require any special privileges. For example, the routing-based anycast implementation requires that special routes to anycast nodes are advertised in the Internet. This means in particular that anybody deploying routing-based anycast needs to control routers able to inject new routes, which might already be beyond the reach of most regular Internet users.

3 Alternative Implementations

A number of systems have been proposed to provide anycast-like functionality. This section analyzes to what extent they meet the requirements discussed in Section 2.

Proxying Frontend In the most straightforward approach, one can use a proxying frontend, which would forward client traffic to individual nodes within its anycast group, and whose network address would be advertised as the anycast address of that group [8]. Such a solution offers real-time, fine-grain control over the client traffic and can easily support connection handoffs. However, when used in wide-area setups, frontends tend to become performance bottlenecks, as they limit network bandwidth available to each anycast group and introduce additional latency to the communication between clients and anycast nodes [7].

Client-side Software Another simple anycast implementation relies on the client-side application to manage anycast communication. In that case, the composition of each anycast group must be revealed to the clients, which thus obtain the flexibility of selecting individual anycast nodes, switching between them, and handling their failures [11, 15, 22]. However, disclosing the group composition to the clients introduces the problem of keeping that composition consistent. This might become expensive, especially when the group composition is very dynamic. Also, given that client applications lack detailed information about the load, performance, and availability of individual anycast nodes, they are by nature poorly suited to select the best anycast nodes to communicate with [32]. The last problem is that this approach is only applicable when developing new applications. It does not allow for incorporating anycast into legacy applications that were not initially designed to support anycast communication, such as the Web.

DHTs Anycast functionality can also be implemented with distributed hash tables (DHTs) [10]. Similar to what happens in client-based implementations, DHTs customize the client-side software, which limits their applicability to new systems only. However, instead of letting a single client handle the anycast communication, DHTs organize clients and anycast nodes into overlays enabling the former to route messages to the latter. Routing protocols utilized by DHTs are designed such that messages can be exchanged even in the face of frequent changes in overlay composition. There is also no need to disclose the complete composition to any single client. However, the problem with DHT is that selection of anycast nodes is performed implicitly by the routing protocol, which limits the application's control over the mapping between clients and anycast nodes.

Routing-based Anycast The standard anycast implementation exploits the properties of Internet routing protocols, which enable any unicast IP address to be turned into the anycast address of some anycast group [24]. To this end, every node belonging to a given anycast group attaches the same unicast IP address to its own network interface, and lets the route to that address be propagated by the routing system. This results in advertising the same IP address via multiple routes leading to different anycast nodes. Given that routers automatically select shortest routes to each destination IP address, the unicast IP address effectively turns into an anycast address, and all the traffic sent to that address naturally splits among the anycast nodes.

While routing-based anycast has been used to implement critical applications such as load balancing across root DNS servers, it also has several disadvantages. First, similar to the previous approach, clients are redirected to anycast nodes irrespective of the situation within the anycast group, which leaves that group with no control over client traffic. Second, removing anycast nodes requires routing updates to be propagated, which takes some time during which the affected clients cannot contact the anycast group at all [27]. Third, routes to anycast addresses are difficult to aggregate, which increases the overall number of routes to be processed by routers worldwide [20]. Finally, since client traffic is effectively redirected by third-party routers, on-demand traffic switching between anycast nodes is practically impossible. While some of these limitations have been addressed by various research efforts, the solutions typically require either changes in routing protocols [13, 18, 28], or upgrades to the Internet infrastructure [5, 20, 26, 30].

DNS Redirection Anycast-like communication can also be implemented with DNS. To this end, all the anycast nodes within a given group can share the same DNS name, and the DNS server responsible for that name can split client traffic by returning the IP addresses of different anycast nodes to different clients. This scheme has been successfully employed by content delivery networks, as it integrates transparently into the Internet communication model, exploits the scalability of DNS, and provides fairly good control over client redirection [12]. Some research projects have even proposed to extend DNS with advanced anycast functions such as performance-based selection of anycast nodes [34]. However, DNS caching can severely delay updating the redirection mappings, as many DNS servers are configured to ignore short TTL values. This limits the applicability of DNS-based anycast to very stable systems where ungraceful node departures never occur. Also, since client applications typically do not re-validate previously resolved DNS names each time they access an anycast group, switching between anycast nodes is not possible until the DNS name expires and is resolved anew.

Discussion The properties of all the above implementations are summarized in Table 1. Three stars are given to implementations that meet a specific requirement well, two stars mean limited support, and one star means no support whatsoever. As can be observed, each implementation fails to meet at least one requirement completely. This might be why none of them is totally suitable for real large-scale applications, and why systems like Akamai or Google use complex combinations of multiple techniques to implement efficient and reliable anycasting [4, 6]. On the other hand, we believe that a single good implementation must meet all the requirements at the same time. The following section discusses how such anycasting can be implemented using a small set of standard techniques adopted from mobile communication.

4 Architecture

We propose to implement anycast free of all the previously discussed limitations by means of address-translation capabilities provided by the Mobile IPv6 protocol. These capabilities have originally been introduced to enable communication with mobile nodes while they move among various networks. However, we demonstrate that one can also exploit these capabilities to implement anycasting.

The general idea is to present an anycast group to its clients as a single mobile node. The anycast functionality is then implemented by informing each client that this (fictitious) mobile node has moved to the location of the actual anycast node the client is going to communicate with. Similar to what happens in mobile environments, announcing the movement causes the client to redirect all its traffic targeting the mobile node to the new location while keeping the movement transparent to the client applications. This effectively enables the anycast nodes to jointly service their clients via a single anycast address.

The following section discusses some basic aspects of Mobile IPv6, which is the standard protocol designed for mobile communication. Then, we show how selected functions of Mobile IPv6 can be used to implement versatile anycast.

4.1 Mobile IPv6

Mobile IPv6 (MIPv6) consists of a set of extensions to the IPv6 protocol [19]. MIPv6 has been proposed to enable any *IPv6 mobile node* (MN) to be reached by any other *correspondent node* (CN), even if the MN is temporarily away from its usual location. MIPv6 assumes that each MN belongs to one home network, which contains at least one MIPv6-enabled router capable of serving as a *home agent* (HA). Such an HA acts as a representative for the MN while it is away.

To allow one to reach an MN while it is away from home and connected to some visited network, MIPv6 distinguishes between two types of addresses that are assigned to MNs. The *home address* identifies an MN in its home network and never changes. An MN can always be reached at its home address. An MN can also have a *care-of address*, which is obtained from a visited network when the MN moves to that network. The care-of address represents the current physical network attachment of the MN and can change as the MN moves among various networks. The MN reports all its care-of addresses to its HA.

The goal of MIPv6 is to ensure uninterrupted communication with MNs via their home addresses and independently of their current network attachment. To this end, MIPv6 provides two mechanisms to communicate with MNs that are away from home. The first mechanism is *tunneling*, by which the HA transparently tunnels the traffic targeting the home address of an MN to the care-of address of that node (see Figure 1a).

The advantage of tunneling is that it is totally transparent to the CNs. Hence, no MIPv6 support is required from any node other than the MN and its HA. However, tunneling can also lead to two problems. First, if many MNs from the same home network are away, then their shared HA can become a bottleneck. Also, if the distance between an MN and its home network is large, then tunneling can introduce significant communication latency. These two problems are addressed by the second MIPv6 communication mechanism, called *route optimization*. It enables an MN to reveal its care-of address to any CN to allow direct communication (see Figure 1b).

	Efficient Communication	Easy Deployment	Traffic Control	High-Churn Tolerance	Handoff Support
Proxying Frontend	*	***	***	***	***
Client-side Software	***	*	*	***	***
DHTs	**	*	*	***	*
Routing-based Anycast	***	**	*	*	*
DNS Redirection	***	***	**	*	*

Table 1. Comparison of alternative anycast implementations

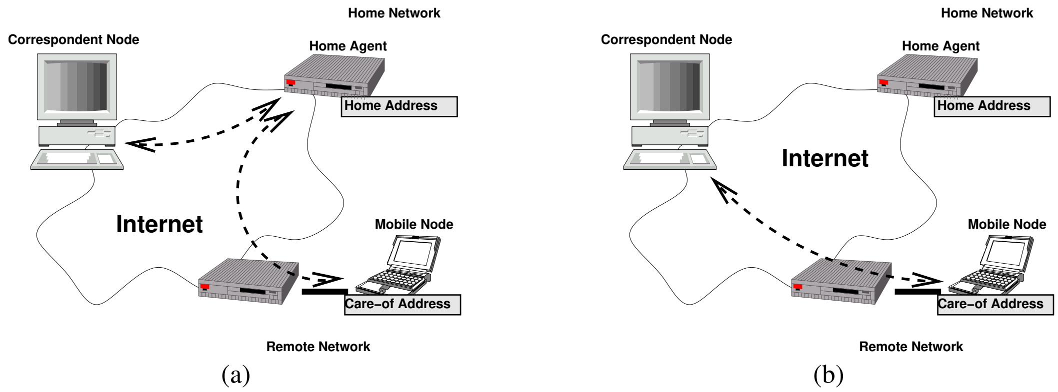

Figure 1. Communication in Mobile IPv6: tunneling (a), and route optimization (b)

Revealing the care-of address causes the CN to create a translation binding between the home- and care-of addresses of an MN. The binding allows the CN to translate between home- and care-of address in the incoming and outgoing traffic, which enables the CN to communicate with the MN directly at its care-of address. This eliminates the latency introduced by tunneling, and offloads the HA.

Route optimization is slightly less transparent than tunneling, as the IP layer at the CN is aware of the current physical attachment of the MN. However, that information is confined inside the IP layer, which effectively hides care-of addresses from higher-level protocols such as TCP and UDP. As a consequence, these protocols use only the home address of an MN and the changes in the MN's location remain transparent to applications running on CNs.

4.2 Versatile Anycast

Our anycast implementation exploits the fact that Mobile IPv6 decouples home- and care-of addresses, effectively allowing for the traffic directed to the former to be transparently redirected to the latter. This comes close to the anycast communication model, in which traffic sent to the anycast address of an anycast group is routed to the interface of some anycast node within that group.

Recall that our solution causes each anycast group to appear to its clients as an MN. The anycast address X of that group then becomes the home address of that fictitious MN. The addresses of anycast nodes within the group, in turn, act as care-of addresses to which the traffic can be redirected. By disclosing different care-of addresses to different clients, the anycast group can convince different clients that the MN has moved to different locations (see Figure 2). Note that the client's higher (transport and application) layers retain the illusion that they communicate with the one and only node holding address X, as the translation between home- and care-of addresses is confined in the network layer.

We implement the above communication model in two steps. First, we make sure that any traffic targeting the anycast address reaches one given anycast node within the respective anycast group. Second, we enable that node to transparently handoff clients to other anycast nodes within the group. Realizing these two steps allows us to implement versatile anycast, as we explain next.

4.2.1 Anycast Address Implementation

Constructing an anycast group requires creating its anycast address first. Such an address should be independent of the group composition, as the composition may change at any moment. We achieve this independence in two stages. First, we allow the anycast address to be provided by any anycast

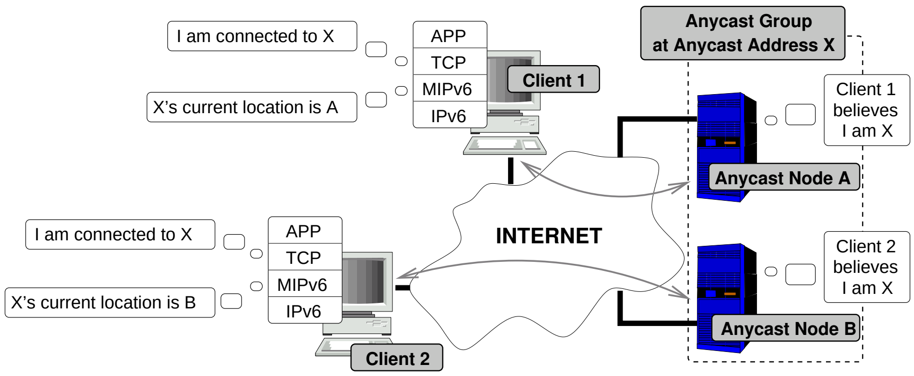

Figure 2. Communication with an anycast group

node within the group, as IPv6 enables any node to generate new IP addresses and attach them to its network interface. Second, we ensure that the anycast address remains valid despite changes in the group composition by allowing it to be *taken over* by any other anycast node as necessary. We refer to the anycast node holding the anycast address of its group as a *contact node*.

To enable the anycast group to move its anycast address at will, the contact node registers that address with its HA, which results in a secret key being shared between the contact node and the HA. The contact node shares that key with some *backup nodes* within the group so that each of them can impersonate the contact node. Impersonating enables each backup node to take over the anycast address once the contact node has left the group, which causes the HA to tunnel all the traffic targeting the anycast address to the backup node. Doing so preserves the reachability of the anycast address as all the traffic addressed to the anycast group keeps on reaching some anycast node.

Although the anycast address is now stable, the performance of anycast communication might still turn out to be poor because extensive tunneling to the new contact node can overload the home agent and introduce communication latency. These limitations are addressed by route optimization wherein the care-of address of an MN is revealed to a CN, allowing for direct communication between them. Since each anycast group appears to its clients and HAs as a single regular MN, it can also use route optimization, causing the clients to communicate directly with the contact node using its actual address. This results in the performance of anycast communication remaining optimal.

Note that the anycast group can prevent the contact node from becoming a potential single point of failure by providing multiple anycast addresses and registering them in the DNS. In that case, different anycast addresses can be handled by different anycast nodes, each acting as a contact node for its respective anycast address. Since these addresses never change, they can safely be registered in the DNS for a long time. Further details of our anycast address implementation can be found in the accompanying technical report [31].

4.2.2 Anycast Traffic Handoff

Our implementation of the anycast address ensures that all the client traffic reaches the contact node. However, this node should not handle all the traffic by itself. It therefore needs a mechanism that allows it to transparently handoff the traffic to other anycast nodes, which later may transparently hand it off again. We refer to the anycast node that hands off a client as a *donor*, and to the anycast node that takes over the client as an *acceptor*.

Recall that address translation in MIPv6 is performed according to bindings created during MIPv6 route optimization. As we discussed in the previous section, anycast groups already exploit this mechanism to establish direct communication between contact nodes and their clients. However, since route optimizations are performed separately for each client, the anycast group can also use them to hand off individual clients between any pair of anycast nodes. To this end, the anycast group carefully mimics the signaling of a mobile node performing route optimization.

Switching the network traffic alone might not be enough, as many applications communicate with their clients using stateful connections such as TCP. In that case, the donor must provide the acceptor with the state of all the network connections opened by the client, so that the acceptor can continue to communicate with the client using these connections and does not reset them. Depending on the application, the same might hold for the application-level state of the client. Our anycast implementation provides anycast nodes with the ability to exchange all such state information as necessary. Further details of how this is done can be found in [31].

4.3 Discussion

Our anycast implementation meets all the base requirements described in Section 2. First, it meets both functional requirements, as it allows one to organize widely distributed nodes into anycast groups addressable by anycast addresses indistinguishable from regular IPv6 addresses.

Second, since clients communicate directly with individual anycast nodes, the only overhead of anycasting is the small initial delay caused by client handoff. Our measurements performed on a prototype anycast testbed confirm our delay analysis and indicate that this delay equals $6 * L_{CS} + 3 * L_{SS}$, where L_{CS} is the one-way latency between the client and the contact node, and L_{SS} is the one-way latency between the contact node and the anycast node that ultimately services the client. When handing off clients that already communicate with some anycast node, some parts of handoff signaling can be performed in advance [31]. This allows the handoff delay to be reduced to $4 * L_{CS} + L_{SS}$.

Third, even though switching client traffic using Mobile IPv6 requires the contact node to maintain some state about each client, the size of that state is very small. The state essentially consists of a single integer number denoting how many route optimizations have been performed between a given client and the anycast group. MIPv6 uses that number to order its messages properly. The small state size enables each contact node to handle a huge number of clients, which makes our anycast implementation extremely scalable.

Finally, our implementation does not require any special changes to the Internet infrastructure as Mobile IPv6 is a standard Internet protocol. Given that MIPv6 has already been implemented in many popular operating systems, it should be easy to exploit our anycast implementation once IPv6 is widely adopted as well [1, 2].

Apart from meeting the base requirements, our anycast implementation also meets the three additional requirements. First, the contact node can handoff clients to other anycast nodes according to any set of metrics. This provides the anycast group with full control over how the client traffic is split among anycast nodes.

Second, the ability to handoff clients at will enables anycast nodes to leave the anycast group at any moment, as all the clients serviced by these nodes can be transparently taken over by any other anycast nodes remaining in the group. This makes anycast groups tolerant to high churn of anycast nodes, allowing anycast groups to be formed in highly dynamic environments such as peer-to-peer overlays.

Third, our anycast implementation causes each client to communicate with its specific anycast node, whose address is revealed to the client during route optimization. This enables clients to communicate with anycast groups using popular stateful protocols such as TCP.

5 Conclusion

In this paper, we have identified a number of limitations specific to existing anycast implementations. We have proposed how all these limitations can be addressed by a single efficient implementation based on Mobile IPv6, so that anycasting can be widely adopted by contemporary distributed systems. Our solution exploits address-translation mechanisms originally introduced to enable communication with mobile nodes. We have demonstrated that these mechanisms can also be used to implement anycast. The only overhead of anycasting is the short delay caused by initial traffic switching, which is a linear function of network latencies between the nodes involved. We exploit our anycast implementation to develop ad hoc distributed servers that preserve their contact address despite rapid changes in their composition [31].

References

[1] MIPL – Mobile IPv6 for Linux. `http://www.mobile-ipv6.org/`.

[2] Mobile IPv6 Systems Research Lab. `http://www.mobileipv6.net/`.

[3] A. Acharya and A. Shaikh. Using Mobility Support for Request-Routing in IPv6 CDNs. In *7th Web Caching Workshop*, Aug. 2002.

[4] M. Afergan, J. Wein, and A. LaMeyer. Experience with some Principles for Building an Internet-Scale Reliable System. In *WORLDS*, Dec. 2005.

[5] H. Ballani and P. Francis. Towards a Global IP Anycast Service. In *SIGCOMM*, Aug. 2005.

[6] L. A. Barroso, J. Dean, and U. Holzle. Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro*, 23(2), 2003.

[7] E. Brewer. Lessons from giant-scale services. *IEEE Internet Computing*, 5(4), 2001.

[8] V. Cardellini, E. Casalicchio, M. Colajanni, and P. Yu. The State of the Art in Locally Distributed Web-Server Systems. *ACM Computing Surveys*, 34(2), June 2002.

[9] V. Cardellini, M. Colajanni, and P. S. Yu. Request Redirection Algorithms for Distributed Web Systems. *IEEE Transactions on Parallel and Distributed Systems*, 14(4), Apr. 2003.

[10] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. Scalable Application-Level Anycast for Highly Dynamic Groups. In *International Workshop on Networked Group Communication*, Sept. 2003.

[11] M. Conti, E. Gregori, and W. Lapenna. Replicated Web Services: A Comparative Analysis of Client-Based Content Delivery Policies. In *Networking 2002 Workshops*, May 2002.

[12] J. Dilley, B. Maggs, J. Parikh, H. Prokop, R. Sitaraman, and B. Weihl. Globally Distributed Content Delivery. *IEEE Internet Computing*, 6(5), Sept. 2002.

[13] S. Doi, S. Ata, H. Kitamura, and M. Murata. Design, Implementation and Evaluation of Routing Protocols for IPv6

Anycast Communication. In *IEEE 19th International Conference on Advanced Information Networking and Applications*, Mar. 2005.

[14] R. Engel, V. Peris, E. Basturk, V. Peris, and D. Saha. Using IP Anycast for Load Distribution and Server Location. In *The 3rd Global Internet Mini-Conference*, Nov. 1998.

[15] Z. Fei, S. Bhattacharjee, E. W. Zegura, and M. H. Ammar. A Novel Server Selection Technique for Improving the Response Time of a Replicated Service. In *INFOCOM*, Mar. 1998.

[16] B. Haberman and E. Nordmark. IPv6 Anycast Binding using Return Routability. Internet Draft, Oct. 2002.

[17] T. Hardie. Distributing Authoritative Name Servers via Shared Unicast Addresses. RFC 3258, Apr. 2002.

[18] W. Jia, D. Xuan, and W. Zhao. Integrated Routing Algorithms for Anycast Messages. *IEEE Communications*, Jan. 2000.

[19] D. Johnson, C. Perkins, and J. Arkko. Mobility Support in IPv6. RFC 3775, June 2004.

[20] D. Katabi and J. Wroclawski. A Framework for Scalable Global IP-anycast (GIA). In *SIGCOMM*, Aug. 2000.

[21] D. Kim, D. Meyer, H. Kilmer, and D. Farinacci. Anycast Rendevous Point (RP) mechanism using Protocol Independent Multicast (PIM) and Multicast Source Discovery Protocol (MSDP). RFC 3446, Jan. 2003.

[22] M. Oe and S. Yamaguchi. Implementation and Evaluation of IPv6 Anycast. In *10th Annual Internet Society Conference*, July 2000.

[23] A. Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, 2001.

[24] C. Partridge, T. Medez, and W. Milliken. Host Anycasting Service. RFC 1546, Nov. 1993.

[25] M. Rabinovich and A. Aggarwal. Radar: A Scalable Architecture for a Global Web Hosting Service. *Computer Networks*, 31(11–16), 1999.

[26] P. Rodriguez and S. Sibal. SPREAD: Scalable Platform for Reliable and Efficient Automated Distribution. *Computer Networks*, 33(1–6), 2000.

[27] S. Sarat, V. Pappas, and A. Terzis. On the Use of Anycast in DNS. In *International Conference on Measurements and Modeling of Computer Systems*, June 2005.

[28] M. Shand and M. Thomas. Multi-homed Host Support in IPv6. Internet Draft, June 1997.

[29] A. Snoeren, D. Andersen, and H. Balakrishnan. Fine-Grained Failover Using Connection Migration. In *3rd USENIX Symposium on Internet Technologies and Systems*, Mar. 2001.

[30] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. In *SIGCOMM*, Aug. 2002.

[31] M. Szymaniak, G. Pierre, M. Simons-Nikolova, and M. van Steen. A Single-Homed Ad Hoc Distributed Server. Technical Report IR-CS-013, Vrije Universiteit Amsterdam, Mar. 2005.

[32] L. Wang, V. Pai, and L. Peterson. The Effectiveness of Request Redirection on CDN Robustness. In *5th Symposium on Operating System Design and Implementation*, Dec. 2002.

[33] S. Weber and L. Cheng. A Survey of Anycast in IPv6 Networks. *IEEE Communications*, Jan. 2004.

[34] 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*, 8(4), 2000.