Anycast in Locality-Aware Peer-to-Peer Overlay Networks

Rongmei Zhang and Y. Charlie Hu

Purdue University, West Lafayette IN 47907, USA
{rongmei, ychu}@purdue.edu

Abstract. This paper advocates implementing anycast in peer-to-peer (p2p) overlay networks. We argue that anycast in p2p overlays (exemplified by Pastry, Tapestry, Chord, CAN) combines the advantages of IP anycast and existing application-layer anycast services. We show that anycast can leverage the locality-awareness embedded in existing p2p overlays. The locality-awareness of the p2p routing substrates can be extended to support anycast and anycast is achieved as the result of the enhanced locality-aware routing. We have implemented anycast in Pastry, and experiments confirm that with high probability, a message addressed to an anycast group is routed to the closest node in the group according to the proximity metric. We also evaluate the performance of anycast using a realistic failure trace and the results show that our implementation of anycast is resilient to node failures.

1 Introduction

Anycast for IP was first introduced in RFC 1546 [1]. A message addressed to an anycast address is routed by the network to at least one of the servers that accept messages for that anycast address (see Fig. 1). IP anycast can provide automatic service discovery in the Internet. By assigning the same anycast address to replicated FTP servers, users can download from the closest server without manually choosing from a list of mirrors. IP anycast can also support host autoconfiguration by assigning an anycast address to the DNS service; after moving to a new network, a host can continue to contact the DNS anycast address instead of being re-configured with the new local DNS server.

Application-layer anycast has also been proposed [2–4]. While IP anycast relies on network routers to find the closest receiver, application-layer anycast involves anycast address resolvers to obtain application specific measurements.

The drawbacks of both IP anycast and application-layer anycast have been well known. Anycast does not comply with the hierarchy of the Internet and thus IP anycast is difficult to scale. In [5], Katabi and Wroclawski show that it is possible to provide scalable global IP anycast service. However, IP anycast needs support from network routers and its wide usage relies on wide deployment of supporting routers. Application-layer anycast requires the support of Internet distance services and scalable network measurement is still under continuous research efforts.

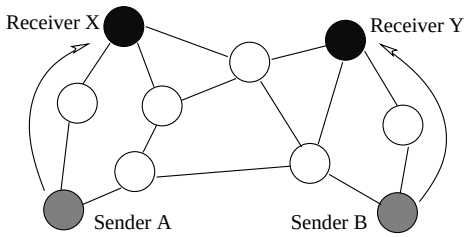


Fig. 1. Illustration of anycast. Sender A and sender B are sending to the same anycast group and their messages are routed to the closest group members receiver X and receiver Y respectively.

In the last few years peer-to-peer systems have gained popularity, from the pioneering Napster [6], Gnutella [7], and FreeNet [8] to the second-generation systems such as CAN [9], Chord [10], Pastry [11], and Tapestry [12]. These second-generation systems all build self-organizing and decentralized overlay networks. Unicast in these systems has been extensively studied. The unicast routing algorithms are highly scalable and efficient; routing is typically finished within a small number of overlay hops. Multicast on top of these p2p overlay have also been studied [13–16]. This paper advocates integrating anycast into p2p overlay networks to leverage their inherent scalability and efficiency.

Implementing anycast in p2p overlays has several advantages. First, each node in the p2p network volunteers as a router and maintains a routing table. Thus implementing anycast in p2p networks has the advantage of easy access to the "routers" while IP routers are more difficult to access and manipulate. Second, p2p overlay networks employ a flat address space by hiding the hierarchical structure and other details of the underlying Internet, which simplifies the implementations. Third, p2p overlays are highly scalable from their decentralized organization. Routing in a p2p overlay is usually completed within $logN$ hops, and by taking into account network proximity during p2p routing (called locality-aware routing), very low routing stretch and network stress are incurred. Therefore p2p overlays are deployable in large networks. Fourth, as in previous application-level anycast systems, p2p overlays can support multiple proximity metrics for anycast such as routing delay or network bandwidth.

A major issue in implementing anycast is how to route to the closest group member in an anycast group. We argue that a locality-aware p2p overlay provides an ideal environment for implementing anycast. The key observation is that locality-aware p2p overlays exploit locality in the underlying physical network in performing routing in the overlays [17, 18], and this "locality-awareness" can be naturally exploited in implementing anycast, which by definition routes each message to the closest node among all candidate nodes.

The rest of the paper is organized as follows. Section 2 gives an overview of one of the locality-aware structured p2p overlays, Pastry. Section 3 describes how to integrate anycast into Pastry. Section 4 presents simulation results showing the effectiveness of anycast in finding the closest group member. Section 5 discusses related work and Section 6 concludes the paper.

2 Background

In this section, we give a slightly detailed description of Pastry [11, 19], since we propose to support anycast by extending Pastry's unicast substrate. Pastry is a scalable, fault-tolerant, peer-to-peer substrate. Each Pastry node has a unique, uniformly randomly assigned *nodeId* (node identifier) in a circular 128-bit identifier space. Given a 128-bit key, Pastry routes the associated message towards the live node whose nodeId is numerically closest to the key.

Routing state For the purpose of routing, nodeIds and keys are thought of as a sequence of digits in base 2^b. A node's routing table is organized into $128/b$ rows and 2^b columns. The 2^b entries in row n of the routing table contain the IP addresses of nodes whose nodeIds share the first n digits with the present node's nodeId; the $(n+1)$th nodeId digit of the node in column m of row n equals m. A routing table entry is left empty if no node with the appropriate nodeId prefix is known.

Each node also maintains a *leaf set*. The leaf set is the set of l nodes with nodeIds that are numerically closest to the present node's nodeId, with $l/2$ larger and $l/2$ smaller nodeIds than the current node's nodeId. The leaf set ensures reliable message delivery and is used to store replicas of application objects.

Routing At each routing step, a node seeks to forward the message to a node whose nodeId shares with the key a prefix that is at least one digit (or b bits) longer than the current node's shared prefix. If no such node can be found in the routing table, the message is forwarded to a node whose nodeId shares a prefix with the key as long as the current node, but is numerically closer to the key than the present node's nodeId.

Node joining A new node with nodeId X joins the network by asking an existing, nearby Pastry node A to route a join message using X as the key. The message is routed to the existing node Z with the nodeId numerically closest to X. Node X then obtains the leaf set from Z and appropriate routing table entries from nodes encountered along the path from A to Z. After initializing its state, node X notifies other nodes that need to know of its arrival and thereby updates their routing states accordingly.

Routing table maintenance To prevent the deterioration of the locality of routing table entries when the underlying network changes over time, Pastry employs a *routing table maintenance* mechanism. Periodically, each node selects a random entry from each row of its routing table, and requests from the associated node a copy of that node's corresponding routing table row. Each entry from the returned routing table row is then compared to the corresponding entry in the local routing table row. If they differ, the closer node is installed based on the proximity metric.

When a routing table entry is replaced during the maintenance, it can be stored in the routing table as a backup for the primary node. When the primary

node fails, the closest live backup node is used. Storing backup nodes effectively extends the routing table to three-dimensional [19]. In this paper, the routing table entry size is set to 10.

2.1 Locality-aware routing

Through its locality-aware node join process and routing table maintenance mechanism, Pastry maintains a proximity-aware overlay by minimizing the distance, according to a proximity metric such as network delay, to each of the nodes that appear in a node's routing table. More precisely, Pastry ensures the following invariant for each node's routing table:
Proximity invariant: *Each entry in a node X's routing table refers to a node that is near X, according to the proximity metric, among all the live Pastry nodes with the appropriate nodeId prefix.*

Because of the above proximity invariant and prefix-based routing, Pastry routing exhibits *low delay stretch* – the total delay experienced by Pastry routing relative to the delay between the source and destination via the underlying IP routing is usually below two [11, 19].

3 Anycast in locality-aware p2p overlays

In this section, we discuss how to implement anycast in structured p2p overlays. We use Pastry [11] as a concrete example in our description. In principle, anycast can be implemented in any other structured p2p substrate that supports locality-aware routing [18], such as CAN [9], Chord [10], or Tapestry [12].

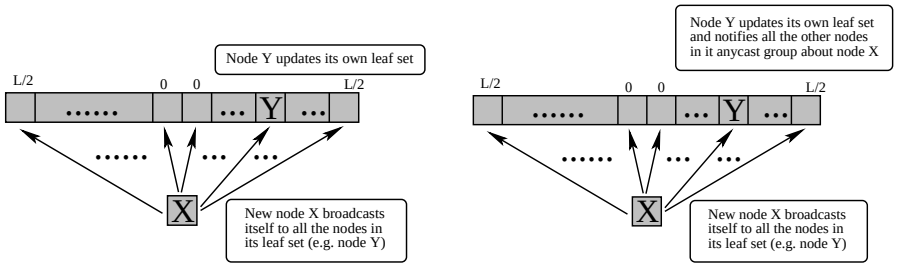
3.1 Overview

Our design of anycast-enabled Pastry extends the original Pastry in two major ways: (1) An (anycast) nodeId can correspond to multiple physical nodes (anycast group members). The anycast group members are required to maintain consistent leaf sets. (2) Leaf sets are maintained based on locality and may be two-dimensional by configuration. As a result, each anycast entry in the leaf set points to nodes close to the local node in the proximity space.

The above extensions effectively leverage Pastry's locality-aware routing to support anycast. In Pastry routing, each overlay hop is either taken from the leaf set or the routing table. Since both are maintained based on the proximity metric in anycast-enabled Pastry, each hop is always close to the current node. As a result, anycast-enabled Pastry always routes a message destined to an anycast group to an anycast group member that is near the source.

3.2 Anycast nodeIds

In structured p2p overlays, nodeIds are identifiers of nodes in the overlay network and are assumed to be unique. Furthermore, the mapping between nodeIds and



(a) Case 1: Node Y has seen the nodeId of node X before and it simply updates its own leaf set according to the proximity metric.

(b) Case 2: Node Y is seeing the nodeId of node X for the first time and updates its own leaf set. If node Y is an anycast node, it also informs all the other nodes in the same anycast group of the arrival of node X.

Fig. 2. The leaf set update process during node join.

IP addresses is assumed to be one-to-one. To incorporate anycast, the above assumption is extended to allow multiple nodes to share the same anycast nodeId. These nodes form an anycast group.

3.3 Node joining

In anycast-enabled Pastry, the routing of a join message and the initialization of the new node's routing state is the same as in the original Pastry. However, the leaf set update process after the node joining is distinctive in two aspects, as shown in Fig. 2.

Locality-aware leaf set update As in the original Pastry, at the end of the join process, the joining node broadcasts its arrival to the nodes in the inherited leaf set so that those nodes can update their leaf sets. In anycast-enabled Pastry, this update takes locality into account. For example, in Fig. 2(a) as the new node X notifies node Y in its inherited leaf set, if node X does not represent a new nodeId but is closer in the proximity space to node Y than the corresponding entry in node Y's current leaf set, node Y replaces that old entry with node X. The replaced node is kept as a backup in case failures should happen.

Maintaining leaf set consistency When anycast is incorporated into Pastry, there are multiple nodes in the overlay network corresponding to an anycast nodeId. By definition, the leaf sets of these nodes should contain the same set of nodeIds. Leaf set consistency is a necessary condition for the correct routing behavior of a Pastry overlay network; it guarantees that a message is always routed to the node whose nodeId is numerically closest to the message key.

Leaf set inconsistency can potentially occur at the end of a node join process when the newly joined node X announces its existence to an anycast node Y in its inherited leaf set and node Y sees the nodeId of X for the first time, as shown in Fig. 2(b). If node Y does not notify other nodes in the same anycast

group, those nodes will not be aware of the existence of the new nodeId of node X. To avoid this inconsistency, upon receiving from node X, node Y informs all the other anycast group members about node X, using a group communication mechanism.

Anycast group communication We use multicast to support group communication within each anycast group. Any prefix-based p2p multicast protocol [13–15] can be used. The multicast address (multicast nodeId) of an anycast group is obtained by a deterministic mapping from the anycast nodeId, for example, by adding 2^{127} to the anycast nodeId (modulo 2^{128}). An update message is propagated from any node to all the other nodes by traversing the multicast tree.

3.4 Locality-aware routing in anycast-enabled Pastry

Anycast-enabled Pastry inherits locality in routing tables from the original Pastry. In addition, it also extends locality to leaf sets.

Extended proximity invariant: *Each entry in a node X's routing table refers to a node that is near X, according to the proximity metric, among all the live Pastry nodes with the appropriate nodeId prefix. Each entry in a node X's leaf set refers to a node that is near X, according to the proximity metric, among all the live Pastry nodes that share the same nodeId.*

When a new node X joins the network, it will be routed to a node Z with the numerically closest nodeId. If node Z belongs to an anycast group (e.g., when node X and Z are from the same anycast group), then node Z is also among the closest to node X in the proximity space among all the existing anycast group members. Assuming the triangle inequality holds in the proximity space, if the nodes in node Z's leaf set are close to node Z, they are also close to node X after node X inherits them as its own leaf set. On the other hand, if node Z has a unique nodeId, the anycast nodes in its inherited leaf set may not be close to node X. In this case, the locality of the inherited leaf sets will be improved over time by the leaf set update procedures including leaf set maintenance (see Section 3.5).

At the end of the join process, nodes from node X's inherited leaf set are notified of its arrival. These nodes are close to node X in the proximity space and may update their own leaf sets based on proximity comparison (Section 3.3). On the other hand, those nodes whose nodeIds fall inside the range of but are not included in node X's leaf set are likely to be distant in the proximity space. Therefore they are not notified of the new node X since they are unlikely to benefit from the update. The exception to this is that when node X represents a new nodeId in the p2p network, all anycast members of each anycast node in the inherited leaf set will be notified (Section 3.3).

In summary, as a result of the node join protocol and the leaf set updating procedures, the *proximity invariant* of Pastry routing is automatically extended to include the leaf set of each node when anycast is integrated into the overlay network. Since the locality in routing tables are initialized and maintained as in

the original Pastry, an anycast nodeId in the routing table also points to a close node in the corresponding anycast group according to the proximity metric.

3.5 Handling node failures

Node failures in routing tables The anycast-enabled Pastry uses the same reactive procedure as in the original Pastry to lazily repair failed entries in the routing table. The routing algorithm chooses an alternative node to forward the message if the best choice from the routing table is found to have failed. If the down stream node has a routing table entry that matches the next digit of the message key, it automatically informs the upstream node of that entry. In addition, as in the original Pastry, the periodic routing table maintenance also repairs failed entries.

Node failures in leaf sets Anycast-enabled Pastry uses a mechanism called *leaf set maintenance* to repair failed leaf set entries: each live leaf set entry is asked to return its own leaf set and the returned leaf set is merged with the local leaf set. In addition to repairing failed entries, the local leaf set is also updated based on the proximity metric: if both the local entry and the returned entry are alive, the closer one in the proximity space is installed. Similar to routing table maintenance, those nodes that fail in the proximity comparison can be stored in the leaf set as backups to the primary node [1]. Storing backup nodes effectively extends the leaf set to two-dimensional. In this paper, the leaf set entry size is set to 10.

The backups can be used to help routing as well as leaf set maintenance. When the primary node fails, the closest live backup node can be used for routing or requesting leaf set during leaf set maintenance. The above leaf set maintenance can also be invoked periodically to prevent locality deterioration in leaf sets when the network is dynamic. Therefore, the impacts of leaf set maintenance are two fold: first, it effectively recovers leaf sets from node failures; second, it also helps to maintain the locality of leaf sets.

Node failures in multicast trees Node failures in multicast trees used for maintaining consistency among leaf sets of anycast group members are handled by the corresponding multicast protocol. For example, when the Scribe [13] multicast protocol is used, each node probes its parent periodically and re-joins the multicast tree if the current parent is found to have failed. The prefix-based routing guarantees that a valid, i.e., loop free, tree is always formed.

4 Evaluation

We have implemented anycast in FreePastry [20], following the descriptions in the previous section. This section presents the simulation results using a realistic network topology model and a node failure trace.

[1] Backup nodes are also produced during the leaf set update after a new node joins the p2p network (Section 3.3)

4.1 Experiment setup

The experiments are performed on a network with 1050 routers (50 in transit domains and 1000 in stub domains). The router network is generated by GT-ITM using the transit-stub model [21]. End nodes are assigned randomly to the 1000 stub routers with uniform probability. The routing policy weights generated by the GT-ITM generator are used to calculate the shortest path between any two nodes.

The p2p network is formed by 32000 end nodes with unique nodeIds and an additional number of anycast nodes, each belonging to an anycast group. The anycast groups are modeled by the Zipf distribution: $Sub(r) = \lfloor N(r + 1)^{-1.25} + 0.5 \rfloor$, where r is the group rank and $Sub(r)$ stands for the group size. The group rank ranges from 0 to 15 in the simulations. The largest group has 256 members and the smallest has 8. For each group rank, 12 groups with distinct anycast nodeIds are created, and there are 192 anycast groups in total. Group members are randomly selected with uniform probability from the end nodes in the underlying physical network. Overall, there are a total of 7992 anycast nodes in the simulation.

To evaluate the performance of anycast in the presence of failure and recovery of anycast nodes in the p2p network, we use a trace of node arrivals and departures from a study measuring the availability of desktop computers in a corporate network [22]. The original trace contains the liveness states of 65000 nodes over a period of 840 hours. We use 7792 of the nodes during a period of 60 hours. There are an average about 6538 live anycast nodes in the entire p2p network at each hour in the simulation. We assume that nodes fail silently and the failed node is not discovered by another node until that node tries to route through it a message, either a data packet, or a control message as part of the maintenance procedures. We also assume that an anycast node always comes back with the same anycast nodeId.

At each hour in the simulations, 256 messages are sent from random sources among the 32000 non-anycast nodes to each of the 192 anycast groups. We measure the success rate of anycast routing and the routing performance of anycast, such as the relative delay penalty and the number of routing hops. The *success rate* for each message addressed to an anycast group is defined as the percentage rank, according to the proximity to the message source node, of the actual receiving member out of all the members in that anycast group. For example, the closest and second closest group members to the source node out of 5 group members have a rank of 100% and 80%, respectively. The *relative delay penalty* (RDP) is defined as the ratio between the distance traversed in the overlay network and the distance traversed if it were routed directly by the underlying IP network. If the message is forwarded to a node that has failed in the simulation, the round trip delay to the failed node is added to the total delay experienced by the message to account for the effects of node failure.

In our implementation, Scribe [13] is used to implement anycast group communication. Multicast tree maintenance are invoked at each hour to repair multicast trees for anycast groups. Failures in leaf sets are repaired on demand by

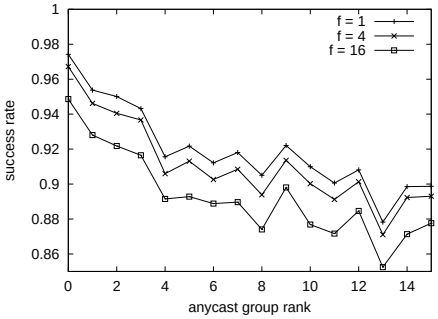


Fig. 3. Success rate of anycast for groups of varying sizes.

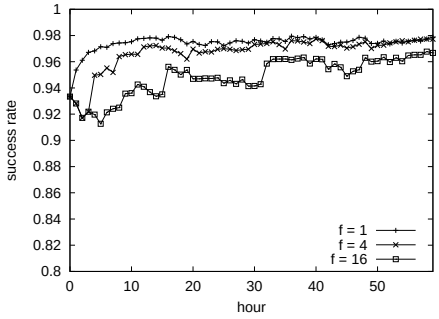


Fig. 4. Success rate of anycast for groups of size 256.

triggering leaf set maintenance. Routing table maintenance is invoked periodically at varying intervals (every 1, 4, and 16 hours, as represented by the symbol f in the figures shown below) to study the correlation between the success rate of anycast and the locality of the p2p network. The nodeId base of Pastry (b) is configured to 2 and the size of the leaf set (l) is set to 16.

4.2 Success rate

Fig. 3 shows the average success rate of anycast for groups of varying sizes (group ranks). It shows the more frequent the routing table maintenance, the higher the success rate. This confirms that the locality of anycast is improved when the locality of routing table entries is improved by routing table maintenance. It also shows that the success rate decreases as the anycast group becomes smaller. This is because the smaller the anycast group, the larger the gap between adjacent ranks of anycast group members.

Fig. 4 shows the distribution of average success rate of anycast routing over the 60 hours of simulation time for groups of 256 nodes (group rank 0). For a fixed maintenance frequency, the success rate improves in the beginning of the simulation and then converges to a steady value. This suggests that the initial locality of the p2p overlay is improved by the maintenance procedures (Section 3.4).

Fig. 5 and Fig. 6 show the cumulative distribution of the success rate of anycast routing for each message and of the average success rate for messages destined to each anycast group, respectively. Under any routing table maintenance frequency, about 80% of all messages have a success rate of about 90%. The average success rates for almost all the anycast groups are above 80%.

The above simulation results suggest that the performance of anycast routing is improved as the locality of the Pastry routing is improved via periodical routing table maintenance. In principle, periodically leaf set maintenance should have similar impacts. However, due to the extremely sparse distribution (less than 1%) of anycast nodeIds within the entire Pastry nodeId space in our simulations, periodical leaf set maintenance does not affect the results as noticeably as routing table maintenance.

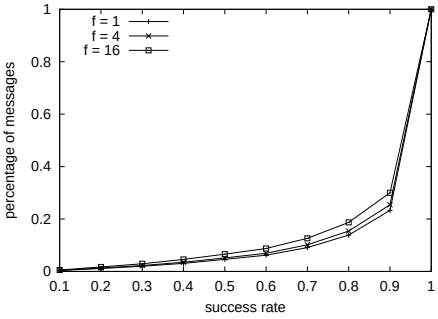


Fig. 5. Cumulative distribution of the success rate of anycast for all messages.

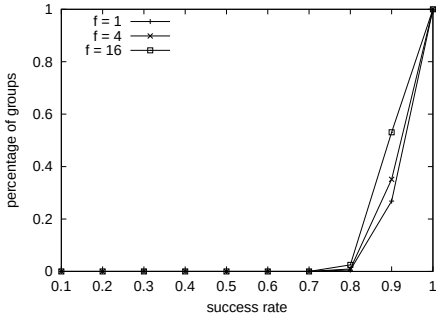


Fig. 6. Cumulative distribution of the average success rate of anycast for each anycast group.

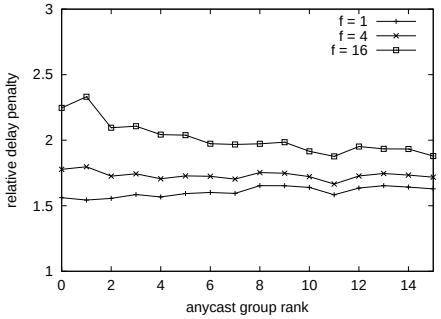


Fig. 7. Relative delay penalty of anycast for groups of varying sizes.

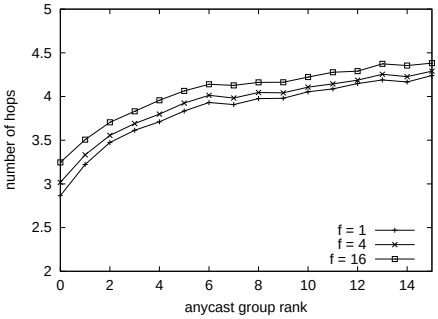


Fig. 8. Number of routing hops of anycast for groups of varying sizes.

Note that our implementation does not always route to the closest node in the proximity space with the appropriate anycast nodeId. This can be explained by the fact that the locality-aware mechanism in Pastry is based on heuristics and thus does not produce perfect routing tables and leaf sets in anycast-enabled Pastry. Building optimal routing tables and leaf sets requires a global view of the p2p network at each node, which is very costly in practice because of the communication overhead. Nevertheless, our implementation of anycast achieves close to optimal performance without incurring excessive communication overhead.

4.3 Routing delay and routing hops

Fig. 7 and Fig. 8 show the average RDP and the average number of p2p routing hops by anycast. First, both metrics are improved by routing table maintenance which improves the locality in routing table entries. Second, the higher the group rank, the fewer the routing hops. This is because the larger the anycast group, the closer the closest member, and the fewer hops it takes to reach that member. Third, RDP largely remains the same for anycast groups of different ranks for $f = 1$ and $f = 4$. This is because RDP is relative to the distance by direct IP routing between the source and the destination. The graduate decrease for

$f = 16$ indicates that the initial locality from node joining is improved gradually at this maintenance frequency.

5 Related work

Anycast was first introduced in RFC 1546 [1] as a means for automatic service discovery and host configuration. IP-anycast has also been proposed to improve IP multicast routing efficiency [23, 24]. Global IP-Anycast (GIA) [5] is an architecture for scalable global IP-anycast in which inter-domain IP-anycast is implemented by edge domains maintaining efficient routes to popular anycast groups and supporting inexpensive routes to unpopular groups. Recent work related to anycast also includes global distance measurement services [25, 26] and server selection techniques based on distance estimation [27].

Application-layer anycast [2] is designed to support server replication and selection without network-layer support, and is provided by anycast domain name resolvers that measure and maintain server performance metrics [3, 4].

Recently, anycast has been proposed as a communication primitive in the Internet Indirection Infrastructure [28]. An anycast group is identified by a k-bit prefix and the remaining bits are used for encoding application-specific preferences. Two possible techniques of encoding location are proposed: zip code or latency based as in [26]. As pointed out in [28], zip code can not always represent network distance accurately. Latency based encoding requires the support of a measurement infrastructure, such as landmark nodes.

6 Conclusions

In this paper, we have shown that anycast can be easily integrated into locality-aware p2p overlays, using Pastry as an example. The resulting anycast-enabled p2p overlay is a more general paradigm for p2p routing since it degenerates into the original "unicast"-only overlay if nodeIds are unique. Our implementation of anycast is effective in locating the closest anycast group member in the proximity space, and retains the low routing penalty of locality-aware p2p overlays. The performance of anycast is closely related to the locality of the p2p networks, and node failures can be tolerated using simple overlay maintenance mechanisms.

Acknowledgment

We thank Peter Druschel and the anonymous reviewers for their helpful comments. This work was supported by an NSF CAREER award (ACI-0238379).

References

1. Partridge, C., Mendez, T., Milliken, W.: Host Anycast Service. RFC 1546. (1993)
2. Bhattacharjee, S., Ammar, M.H., Zegura, E.W., Shah, V., Fei, Z.: Application Layer Anycasting. In Proc. IEEE INFOCOM. (1997)
3. Fei, Z., Bhattacharjee, S., Zegura, E.W., Ammar, M.H.: A Novel Server Selection Technique for Improving the Response Time of a Replicated Service. In Proc. IEEE INFOCOM. (1998)
4. Zegura, E.W., Ammar, M.H., Fei, Z., Bhattacharjee, S.: Application-layer Anycasting: a Server Selection Architecture and Use in a Replicated Web Service. IEEE/ACM Transactions on Networking (2000)

5. Katabi, D., Wroclawski, J.: A Framework for Scalable Global IP-Anycast (GIA). In Proc. ACM SIGCOMM. (2000)
6. Napster. http://www.napster.com/.
7. The Gnutella protocol specification. http://dss.clip2.com/GnutellaProtocol04.pdf. (2000)
8. Clarke, I., Sandberg, O., Wiley, B., Hong, T.W.: Freenet: A Distributed Anonymous Information Storage and Retrieval System. In Workshop on Design Issues in Anonymity and Unobservability. (2000)
9. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Schenker, S.: A Scalable Content-Addressable Network. In Proc. ACM SIGCOMM. (2001)
10. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In Proc. ACM SIGCOMM. (2001)
11. Rowstron, A., Druschel, P.: Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Proc. Middleware. (2001)
12. Zhao, B.Y., Kubiatowicz, J.D., Joseph, A.D.: Tapestry: An Infrastructure for Fault-Resilient Wide-area Location and Routing. Technical Report UCB//CSD-01-1141, U. C. Berkeley (2001)
13. Rowstron, A., Kermarrec, A.M., Castro, M., Druschel, P.: Scribe: The design of a large-scale event notification infrastructure. In Proc. NGC'01. (2001)
14. Zhuang, S.Q., Zhao, B.Y., Joseph, A.D., Katz, R.H., Kubiatowicz, J.: Bayeux: An Architecture for Scalable and Fault-tolerant Wide-Area Data Dissemination. In Proc. NOSSDAV'01. (2001)
15. Zhang, R., Hu, Y.C.: Borg: A hybrid protocol for scalable application-level multicast in peer-to-peer systems. In Proc. NOSSDAV'03. (2003)
16. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Application-level Multicast using Content-Addressable Networks. In Proc. NGC'01. (2001)
17. Ratnasamy, S., Shenker, S., Stoica, I.: Routing Algorithms for DHTs: Some Open Questions. In Proc. IPTPS'02. (2002)
18. Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Exploiting Network Proximity in Distributed Hash Tables. In Proc. FuDiCo. (2002)
19. Castro, M., Druschel, P., Hu, Y.C., Rowstron, A.: Exploiting network proximity in peer-to-peer overlay networks. Technical report MSR-TR-2002-82 (2002)
20. FreePastry. http://www.cs.rice.edu/CS/Systems/Pastry/FreePastry/.
21. Zegura, E., Calvert, K., Bhattacharjee, S.: How to Model an Internetwork. In Proc. IEEE INFOCOM. (1996)
22. Bolosky, W.J., Douceur, J.R., Ely, D., Theimer, M.: Feasibility of a Serverless Distributed File System Deployed on an Existing Set of Desktop PCs. In Proc. SIGMETRICS. (2000)
23. Katabi, D.: The Use of IP-anycast for Building Efficient Multicast Trees. In Proc. Global Internet Symposium. (1999)
24. Kim, D., Meyer, D., Kilmer, H.: Anycast RP mechanism using PIM and MSDP. RFC 3446. (2001)
25. Francis, P., Jamin, S., Jin, C., Jin, Y., Raz, D., Shavitt, Y., Zhang, L.: IDMaps: A Global Internet Host Distance Estimation Service. IEEE/ACM Transactions on Networking. (2001)
26. Ng, T.S.E., Zhang, H.: Predicting Internet Network Distance with Coordinates-Based Approaches. In Proc. IEEE INFOCOM. (2002)
27. Ratnasamy, S., Handley, M., Karp, R., Shenker, S.: Topologically-Aware Overlay Construction and ServerSelection. In Proc. IEEE INFOCOM. (2002)
28. Stoica, I., Adkins, D., Zhaung, S., Shenker, S., Surana, S.: Internet Indirection Infrastructure. In Proc. ACM SIGCOMM. (2002)