Heterogeneity-Aware Peer-to-Peer Multicast

Robbert van Renesse[1] Ken Birman[1] Adrian Bozdog[1]
Dan Dumitriu[2] Manpreet Singh[1] Werner Vogels[1]

[1]Dept. of Computer Science, Cornell University
{rvr,ken,adrianb,manpreet,vogels}@cs.cornell.edu

[2]Sony Corporation
dmd17@cornell.edu

Abstract

A P2P system has to be able to *tolerate*, and where possible, *leverage* heterogeneity [10]. The reasons may be functional, for example, in order to deal with Network Address Translation, or simply to improve performance and robustness. In this position paper we show how SelectCast, an Application-Level Multicast protocol, leverages the information provided by Astrolabe, a P2P system that mines the hosts for information about resources.

1 Introduction

Recent years have shown the development of so-called Peer-to-Peer Protocols (P2PP) that allow a large set of users to pool their personal computers and share resources. The advantages over standard client/server technology are now well-known: P2PP can offer many more resources (in a client/server architecture typically only the server's resources are shared), and, perhaps more importantly, allows control over those resources by the users themselves (even if ultimately many of those resources are not actually owned by the users who contribute them to the P2P system).

Many early P2PPs consider all participating hosts equal, regardless of compute and storage capacity, and regardless of how well they are connected to the other hosts. This policy keeps the P2PP protocols simple, and ensures that all users remain equal in the P2P system. On the negative side, this policy can lead to substantial scalability problems, as has been well documented (*e.g.*, [11]).

In practice, the hosts in a large system are far from equal [10]. They have vastly different CPU and storage capacities, and latency and bandwidth between different pairs of hosts can vary by orders of magnitude [13]. Availability varies widely as some hosts appear and disappear from the network on a regular basis, while others are connected almost continuously. Even the type of connectivity differs, as many hosts are behind firewalls, NAT boxes, or hosted by ISPs that do not allow unsolicited incoming traffic. Protocols like FastTrack [5] recognize these differences, and distinguish between so-called supernodes and regular user nodes. Compared to the original Gnutella protocol these are based on, performance is significantly improved [11, 8], but many of the appeals of true P2PPs are lost in the process as the supernodes effectively act as servers to the user nodes.

Some P2PPs exploit *host proximity* in order to optimize message routing without resorting to dedicated servers (*e.g.*, Pastry [12]). Such P2PPs use pings in order to determine latency or bandwidth of connections, and base routing decisions on these measurements. Thus high-latency or low-bandwidth links can be avoided. Still, such P2PPs can suffer from host churn (rapid changes in membership) or variance in CPU and storage resources.

It comes to mind then, whether P2P systems can use heterogeneity to their advantage [10]. In this position paper, we describe the design of *SelectCast*, a peer-to-peer publish/subscribe routing service, built on *Astrolabe*, a peer-to-peer domain aggregation service. Both SelectCast and Astrolabe are heterogeneity-aware, but at the cost of some increased management overhead when compared to other P2PP protocols. Although these services do not rely on any special-purpose servers, they can leverage asymmetries between hosts and between network connections.

2 Application-Level Multicast

IP-level multicast routing is badly supported in today's Internet, and ISPs are loath to improve this situation. Multicast routing is therefore an important candidate for peer-to-peer solutions, as these do not require the cooperation of ISPs [3].

Many Application-Level Multicast Routing Protocols (ALMRPs) have been designed, and all use tree routing in order to get logarithmic scaling behavior with respect to the number of receivers (assuming the tree has some bounded maximum branching factor and is reasonably well balanced). These protocols put an uneven load on the hosts and networks, as most hosts only receive messages, while some hosts, which we call *routers*, have to forward copies of each message to some set of peers. In an ALMRP capable of filtering, these routers may also have to analyse the content of messages in order to decide what links to forward the messages to. For satisfactory performance, robustness, and scale, it is important to select well-provisioned, dependable hosts for routers.

For efficiency, one would like that the path followed by a message from sender to each receiver follows the path that a point-to-point message would have followed. A simple way to accomplish just this is to have the sender send N point-to-point messages, one to each receiver. But this degenerate two-level tree would not scale, because the load on the sender grows linear with the number of receivers. In order to achieve scalability, the branching factor should not be so large that the routers receive too high a load, nor so small so that the paths followed by messages take too many application-level hops. In addition, the routers have to lie approximately on the point-to-point paths between the sender and the receivers. There are other factors, for example whether latency or bandwidth is more important, that influence what makes a good tree.

Narada [3] is one of the earliest proposed ALMRPs. Narada can achieve performance competitive with IP-Multicast [20]. Narada maintains a mesh of hosts, and then runs a protocol similar to conventional DVRMP to accomplish multicast routing among these hosts. Narada, and many subsequent designs, can best be understood as two protocols layered on top of one another: a protocol that maintains a mesh of hosts, and a multicast routing protocol on top of this mesh.

Other examples of such an organization are Bayeux/Tapestry [22], Scribe/Pastry [2], and a multicast facility in CAN [9]. SelectCast and Astrolabe also fit this description, in which SelectCast is the routing protocol while Astrolabe maintains the mesh. In the sections that follow, we will first describe the Astrolabe and SelectCast services, and will then investigate the ways in which Astrolabe and SelectCast use information about heterogeneity in order to improve scalability, robustness, security, and performance.

3 Astrolabe

The Astrolabe service [17] can best be understood as a peer-to-peer implementation of DNS. Hosts are organized in a domain hierarchy, in which the hosts themselves form the leaf domains. Each domain has a set of attributes (*resource records* using DNS terminology). The main difference with DNS is in how these attributes get updated. The attributes of leaf domains are writable only by their corresponding hosts (although they can do so per request of other hosts, of course). Most strikingly, the attributes of an internal domain are not directly writable, but are generated by (continuously) aggregating the attributes of its child domains. The aggregation is specified by SQL aggregation queries associated with each domain. The SQL queries can be dynamically deployed and updated.

Each host maintains a relational table for each internal domain it is in. This *domain table* has a row for each child domain, and a column for each attribute. One of the columns, *id*, identifies the child domain within its parent. By specifying a path name of *id*s, a domain can be named. Each domain table also has a column that maintains the version of the row, and, most importantly, a column named "contacts." The contacts attribute of a leaf domain is the set of addresses of the corresponding host, while the attribute for an internal domain contains a small subset of addresses of hosts within that domain. To generate such a subset, an SQL aggregation query is used just like for any other attribute. In effect, the SQL query *elects* which hosts to use to represent the domain.

The contacts in a domain's table gossip with one another on behalf of that domain *and* all of its ancestor domains' tables, and keep these tables approximately consistent among those contacts. As each domain runs a separate gossip protocol, the updates received by the contacts of a domain also spread to the other hosts within the domain. All that a host needs to get going (if we ignore Astrolabe's security mechanisms) is its leaf domain's path name and an existing Astrolabe host to start gossiping with. The first is typically configured manually (although a self-configuring option exist, as described in [18]), while the latter can be manually configured or automatically discovered through broadcast and/or multicast searches.

4 SelectCast

In SelectCast, trees of communication links are built that reflect the Astrolabe hierarchy. In each domain, a small set of multicast routers are selected using another SQL aggregation query, in the same way that Astrolabe chooses its contacts (but often using different selection considerations). To broadcast a message, it can simply be sent to one of the routers of the root domain. This router in turn forwards the message to one of the routers in each child domain, and so on. This technique is fault-tolerant, as new routers are automatically selected when current routers fail through Astrolabe's continuous aggregation facilities. However, in order to provide better real-time fault-masking, messages may be forwarded to all routers of all child domains, instead of to just one (as is done also in [14]). In that case, a significant amount of redundant traffic is generated and duplicates should be filtered carefully in order to avoid an exponential explosion of messages.

An "in between" forwarding strategy is also possible that makes better use of available resources. For simplicity, assume there are exactly two routers selected for each (non-leaf) domain. All even messages are disseminated using the first router of each domain, while the odd messages are disseminated using the second routers. Thus, in theory throughput can be doubled (although a significant amount of re-ordering is required). Rather than simply dividing the traffic into disjoint sets, FECs or erasure codes may be used for fault masking, at the expense of increased bandwidth use and CPU overhead.

SelectCast is also able to do publish/subscribe-style filtering by attaching an SQL condition (over domain attributes) onto each message. The routers forward a message only to those child domains for which the condition holds. (The results of such conditions are cached for short periods of time in order to reduce computational overhead significantly.) Topic-based subscriptions are made space-efficient by adding a Bloom filter attribute to each domain (the filters are aggregated by essentially OR-ing the bit masks together). Although less trivial and not yet implemented, content-based subscriptions can also be supported in this framework. In addition to topic- and content-based filtering, other possibilities exist as well. For example, SelectCast supports sending a message to all machines that have a load less than 3. For this, the domains have to have a load attribute that is aggregated by taking the minimum. The message is then forwarded to all domains that have a minimum load less than 3. More usefully, a security notification or even a patch may be sent to, say, all XP machines that have not yet installed SP1. Point-to-point messaging can also be expressed using a simple condition.

5 Heterogeneity-Awareness

The performance and robustness of SelectCast and Astrolabe are significantly influenced by how contacts and routers are elected in domains. Such choices can even determine the difference between the system working and not working. Each contact and router is a record with a set of attributes:

`id`:	the Astrolabe path name of the host;
`addrs`:	the current list of addresses;
`issued`:	time at which the host joined.

Applications can add new attributes at will (for example, connectivity, machine type, operating system, load, storage capacity, etc.). The set of multicast routers for a domain is generated using an (extended) SQL query like:

```
SELECT
    BEST(3, UNION(routers), 'connectivity')
    AS routers
```

This query specifies that we like the use the (at most three) hosts that have best connectivity as routers. The set of gossip contacts for a domain may use:

```
SELECT
    LEAST(3, UNION(contacts), 'issued')
    AS contacts
```

This query specifies that the contacts of the domain should be the three contacts of its child domains that have been operational longest. Such a selection would make sense in a system where node churn is high. Hosts that occasionally join the system would be unlikely to become contacts, and thus create little disturbance.

Different domains can use different queries if they wish, and the SQL queries can be changed at run-time. More sophisticated queries could specify different or additional preferences. For example, a practitioner may prefer one operating system over another, while a theoretician may want to chose contacts that run different operating systems in order to increase failure independence.

Such preferences improve performance and robustness of SelectCast and Astrolabe. Recent performance studies on Emulab, to be published in a forthcoming paper, show that SelectCast's performance on LANs is similar to that of the Yoid ALMRP [6], but in more complicated topologies the performance of SelectCast is significantly more stable than that of Yoid as SelectCast balances the load on routers much better in the higher levels of the hierarchy. (Yoid also uses a combination of a mesh and a tree.)

There may also be strictly functional reasons to prefer one router or contact over another. For example, the messages in SelectCast are disseminated from the root to the leaf nodes. However, certain domains may be separated from the rest of the system by firewalls or NAT boxes, and thus the routers for those domains cannot be contacted by the routers of their parent domains. In order to solve this situation, SelectCast supports "reverse connections," where a router pulls messages from its parent router using HTTP requests, possibly through a web proxy. Astrolabe provides the information necessary for SelectCast routers to recognize if reverse connections are necessary (see [18]). Service Level Agreements may be another source of limitations on what connections may be set up between routers.

There are also restrictions in Astrolabe on which hosts can become contacts. For example, in the secure version of Astrolabe, only some hosts in a domain have the signing key for the domain, and thus only those hosts can be contacts for the domain. Also, addresses behind firewalls and NAT boxes should not be exported by queries like the ones above, and should thus be filtered out [18].

A router or contact may have more than one address. This can be because the host has multiple interface cards, or supports multiple protocols, or utilizes an application-level gateway for messaging through firewalls. Each host maintains, for each peer host it gossips with, statistics about the addresses of the peer host. Initially it selects addresses at random, but as statistics are collected, the host weighs the addresses based on their latencies and starts prefering the low latency addresses, while still occasionally checking the other addresses. Similarly, SelectCast hosts maintain connection statistics to router addresses in order to optimize metrics such as throughput, latency, or jitter.

Although the queries give applications considerable flexibility, the choice of routers is restricted by the shape of the Astrolabe hierarchy. This is a limitation in our ALMRP design, as we may not be able to construct an optimal tree of routers for forwarding messages. Another restriction is due to the limited expressiveness of SQL aggregation queries, which has led us several times to extend the language (for example, with support for Bloom filters and nested records). In spite of these limitations, we find that the amount of flexibility is usually sufficient in practice, if not overwhelming. Nevertheless, we are considering to design and implement dynamic hierarchies for Astrolabe.

6 Other Related Work

There are dozens of ALMRP projects, too numerous to mention them here, but most of which take heterogeneity into account (actually, the ones based on DHTs, such as Bayeux, Scribe, and CAN's multicast protocol, do not take heterogeneity into account). Most of these self-configure, attempting to optimize either bandwidth or latency. None of these have the flexibility in choosing routers that Astrolabe offers. SALM [1] is closest in architecture to SelectCast, but the hierarchy is dynamic and a single router for a domain is elected which has the minimal maximum distance to the other hosts in the domain.

In Brocade [21], a BGP-like two-level DHT routing hierarchy is built using Tapestry at both levels. This greatly improves routing distance over single-level Tapestry, at the cost of extra configuration. SelectCast, in some sense, is a generalization of this idea to multiple levels and multicast.

Both the PAST [12] and CFS [4] P2P file services suggest the notion of virtual nodes to the problem of storage heterogeneity. Basically, a host with lots of disk space poses as multiple smaller nodes, with unfortunately highly correlated failure characteristics.

Many P2PPs use a hash function to assign identifiers to nodes, and claim that this way "nearby" nodes (in the id space) are likely to be diverse and have independent failures. Unfortunately, when applied to many objects replicated on such assumptions, the independence is unlikely to hold for a substantial number of objects. [19] suggests a technique to find independent sets of nodes. Such techniques would be easily exploited in SelectCast and Astrolabe to select failure independent routers and contacts.

7 Conclusion

A P2P system should be aware of heterogeneity in the network. The reasons may be functional, for example, in order to deal with security requirements or Network Address Translation, or simply to improve performance and robustness. In order to be able to adapt well to heterogeneity, information about available resources needs to be gathered. Astrolabe is a P2P service that provides this functionality, and we have shown how SelectCast, and indeed Astrolabe itself, use this in order to optimize runtime operation. We believe this paper's conclusions generalize to other P2P applications.

References

[1] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM*, Pittsburgh, PA, August 2002.

[2] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communications (JSAC)*, 2002.

[3] Y. Chu, S. Rao, and H. Zhang. A case for end system multicast. In *Proceedings of ACM SIGMETRICS*, pages 1–12, Santa Clara, CA, June 2000.

[4] F. Dabek, M.F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In SOSP01 [15], pages 202–215.

[5] Fasttrack. http://www.fasttrack.nu.

[6] P. Francis, S. Ratnasamy, R. Govindan, and C. Alaettinoglu. Yoid project. http://www.icir.org/yoid/.

[7] *Proc. of the First International Workshop on Peer-to-Peer Systems (IPTPS'02)*, Cambridge, MA, March 2002.

[8] H.D. Johansen and D. Johansen. Improving object search using hints, gossip, and supernodes. In SRDS02 [16].

[9] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-Level Multicast using Content-Addressable Networks. In *Proceedings of NGC*, November 2001.

[10] S. Ratnasamy, S. Shenker, and I. Stoica. Routing algorithms for DHTs: Some open questions. In IPTPS02 [7].

[11] J. Ritter. Why Gnutella Can't Scale. No, Really. http://www.darkridge.com/j̃pr5/doc/gnutella.html, February 2001.

[12] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale persistent peer-to-peer storage utility. In SOSP01 [15], pages 188–201.

[13] S. Saroiu, K. Gummadi, and S. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proc. of Multimedia Conferencing and Networking*, San Jose, CA, January 2002.

[14] A. Snoeren, K. Conley, and D.K. Gifford. Mesh-based content routing using XML. In SOSP01 [15], pages 160–173.

[15] *Proc. of the 18th ACM Symp. on Operating Systems Principles (SOSP'01)*, Banff, Canada, October 2001.

[16] *Proc. of the 21st IEEE Symposium on Reliable Distributed Systems (SRDS'02)*, Osaka, Japan, October 2002.

[17] R. van Renesse, K.P. Birman, D. Dumitriu, and W. Vogels. Scalable management and data mining using Astrolabe. In IPTPS02 [7].

[18] R. van Renesse and D. Dumitriu. Collaborative networking in an uncooperative Internet. In SRDS02 [16].

[19] H. Weatherspoon, T. Moscovitz, and J. Kubiatowicz. Introspective failure analysis: Avoiding correlated failures in peer-to-peer systems. In SRDS02 [16].

[20] B. Zhang, S. Jamin, and L. Zhang. Host Multicast: A framework for delivering multicast to end users. In *Proc. of IEEE INFOCOM*, New York, NY, June 2002.

[21] B.Y. Zhao, Y. Duan, L. Huang, A.D. Joseph, and J.D. Kubiatowicz. Brocade: Landmark routing on overlay networks. In IPTPS02 [7].

[22] S. Zhuang, B. Zhao, A. Joseph, R. Katz, and J. Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of the Eleventh International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 2001)*, Port Jefferson, NY, June 2001.