Quasar: A Probabilistic Publish-Subscribe System for Social Networks

Bernard Wong   Saikat Guha
Dept. of Computer Science, Cornell University, Ithaca, NY 14853
bwong@cs.cornell.edu   saikat@cs.cornell.edu

Abstract
Existing peer-to-peer publish-subscribe systems rely on structured-overlays and rendezvous nodes to store and relay group membership information. While conceptually simple, this design incurs the significant cost of creating and maintaining rigid-structures and introduces hotspots in the system at nodes that are neither publishers nor subscribers. In this paper, we introduce Quasar, a rendezvous-less probabilistic publish-subscribe system that caters to the specific needs of social networks. It is designed to handle social networks of many groups; on the order of the number of users in the system. It creates a routing infrastructure based on the proactive dissemination of highly aggregated routing vectors to provide anycast-like directed walks in the overlay. This primitive, when coupled with a novel mechanism for dynamically negating routes, enables scalable and efficient group-multicast that obviates the need for structure and rendezvous nodes. We examine the feasibility of this approach and show in a large-scale simulation that the system is scalable and efficient.

1 INTRODUCTION

The recent advent of social-networking communities such as MySpace, Facebook, and Twitter has changed many of the social aspects of communication on the Internet and has, to a degree, disrupted the communication patterns we are used to seeing between users. Social networks provide an avenue for friends and people that share a common interest to not only communicate, but to be kept abreast of their minute-to-minute activities.

In contrast to person-to-person communication systems that have predominated, such as Instant Messaging and E-mail, or group communication systems where individuals subscribe to relatively few groups or receive a low volume of messages, such as Usenet, mailing-lists, and blogs, social networks have unique requirements that make them much more demanding to the underlying communication system. In social networks, it is common for people to have hundreds of friends, and be a part of many communities containing thousands of users. Unlike RSS, where a single publisher originates every message, messages in social networks can originate from any user that must be delivered in a timely manner.

Currently, the popular social-networking systems are centralized with each network controlled by a single entity. But as the demand for social-networking systems grows, both privacy and scalability concerns associated with centralized systems make them undesirable to the end-users. The privacy concerns are clear; an unwarranted amount of trust is placed on these centralized entities to not reveal or take advantage of sensitive information. Poor scalability leads to additional cost which, in today's Internet economy, results in exposing the users to additional advertisements.

In this paper, we outline Quasar, a scalable, peer-to-peer publish-subscribe system that caters to the specific needs of social networks. It is designed to handle social networks with many groups; on the order of the number of users in the network. Nodes maintain state that is constant in the number of overlay links and independent of the number of groups in the system. The system is agnostic to the underlying overlay, allowing the social network designers to decide the type of overlay that best suits their operational needs.

The key mechanism in Quasar is a rendezvous-less event routing infrastructure for performing scalable publish-subscribe to specific groups without requiring per-group state or specific structure in the underlying overlay. To accomplish this, the routing infrastructure provides an anycast-like primitive, allowing messages to be probabilistically routed to a nearby group member by specifying the group as the destination. However, unlike in IP anycast where a group member anycasting to its own group will have the packets routed back to itself, thus making IP anycast unsuitable as a primitive for group communication, Quasar introduces a novel mechanism to avoid these self-loops.

This probabilistic routing mechanism combines proactive dissemination of vectors throughout the network that point in the direction of nearby group members, with a reactive lookup that ensures diversity in selecting members. Group members create local gradients of aggregated vectors or "gravity wells", that attract messages traversing the region in the overlay. A message can selectively negate the effects of gravity wells from specific nodes, allowing fine-grained control over route diversity.

Quasar's design obviates the need for rendezvous nodes and overlay structure that encode group membership information, which is the prevalent design of modern publish-subscribe systems, as messages are directly routed to nearby group members. The absence of rendezvous nodes removes a layer of indirection, and eliminates potential hotspots, while the absence of structure reduces complexity and the susceptibility to churn.

Overall, this paper makes three contributions. First, it introduces a scale-free, anycast-like, rendezvous-less routing primitive for overlay networks. Second, it describes a novel publish-subscribe system that explores a new point in the design space with benefits that cater to the requirements of social-networks. Finally, it shows the system is practical, efficient and hotspot-free in both a large-scale simulation consisting of a social network of 32,000 nodes, and a smaller simulation parameterized with MySpace profile data of over 10,000 users' friends lists and communities.

2 SYSTEM ARCHITECTURE

The most common approach to building publish-subscribe systems is to designate a "rendezvous node" for each group. In DHT-based systems, for instance, the rendezvous node of a group corresponds to the node with the ID closest to the group ID, where IDs are typically generated from the hashes of the node and group names.

While conceptually simple, there are two practical problems with this approach. First, the rendezvous node for a group may be chosen from non-group members, as node selection is based solely on hash proximity. This creates an unnatural trust relationship between a group and its rendezvous. Second, rendezvous nodes of highly-active groups become overlay hotspots. Systems such as Corona [10] and Feedtree [12] address this problem by constructing a replication tree for each group. However, these systems are fragile in the face of node-churn.

These problems led us to avoid the use of rendezvous nodes in our design. Instead, Quasar uses an anycast-like routing for message dissemination. Each message published by a group member is routed to other members without a-priori knowledge of the group membership.

There are a number of different ways to provide such an anycast-like primitive. At one end of the design spectrum, the publisher floods the message along overlay nodes, and recipients discard unwanted messages; this clearly does not scale as messages quickly saturate the network. At the other end, nodes can be made into group aware routers, with routing information for finding the closest group member of every group. This too, does not scale as the per-node storage cost grows linearly with the number of groups in the system. Furthermore, it leads to a system with a high sensitivity to churn, where local events require global route convergence. Quasars uses a hybrid routing approach that exploits this communication-space trade-off by making use of both proactive and reactive routing, in addition to introducing the use of dynamic negative information to avoid loops.

2.1 Proactive Dissemination and Directed Walks

In Quasar, group members install a collection of group-specific routing vectors in nearby overlay neighbors up to a few hops away. Members publish messages to the group by issuing multiple copies of the message in random directions along the overlay. If a message encounters a gradient at a node, it is routed to the member that installed the gradient in a directed walk. A real-world analogy to astronomy is illustrative. Each group member is a black-hole that creates a gravity well around it. Messages are beams of light. Even without knowing the exact location of a black-hole, a beam of light traveling roughly in the direction of the black-hole gets sucked into the gravity well.

The per-node storage requirement for this group-based routing scheme is quite modest. Each overlay node needs to induce gravity to nearby nodes that reflects its group membership. The number of gravity wells a node is part of, therefore, is proportional to the density of nodes in its vicinity and the groups they are members of; further, aggregation is used to reduced the storage requirements. While any number of aggregation methods can be used, we use attenuated bloom filters [8]. This allows the per-node storage to be proportional only to the degree of a node — a constant in small-world networks [7].

The routing performance scales with the group size and the size of the gravity wells in relation to the diameter of the network. The more group members in the system, and the deeper into the network their interests are disseminated, the easier it is for Quasar to discover the full group membership and select short paths to them. In social networks, where the diameters tend to be nearly constant, Quasars allows the network operator to select the degree of dissemination that suits the application.

2.2 Negative Information

While the above routing mechanism has attractive scalability properties, trivially extending it to provide many-to-many routing results in routing loops, including self-loops. In essence, because each group member installs a gravity well to receive messages for the group, nearby nodes route messages sent by that member back to itself; this is the same problem that prevents IP anycast

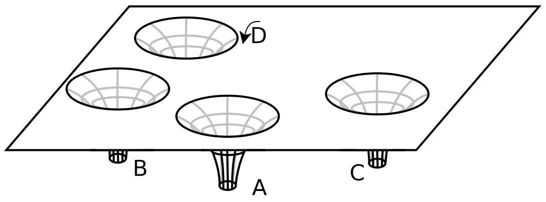

Figure 1: Gravity wells created for group G by group members A–D during the join process. All wells can be indexed using G; individual wells can be indexed using the corresponding node ID. The diameter of the well is $2K$, where K is the depth of the attenuated bloom filter. Gravity wells may overlap (not shown for clarity).

Algorithm 1 SENDUPDATES(N, B, K, T)

```
Require: N: Node ID of the local node
Require: B: Bloom filter size
Require: K: Attenuated bloom filter depth
Require: T: Freshness TTL
Ensure: N's neighbors are notified of groups reachable through N
 1: H ← EMPTYATTENUATEDBLOOMFILTER(B, K)
 2: for all G in SUBSCRIBEDGROUPS(N) do
 3:     H[0].INSERT(G)
 4: H[0].INSERT(N) {used for negative information}
 5: for all O in OVERLAYNEIGHBORS(N) do
 6:     if LASTUPDATETIME(O) < NOW() - T then
 7:         continue
 8:     F ← GETATTENUATEDFILTERFOR(O)
 9:     for all L in 1...K do
10:         H[L].MERGE(F[L-1]) {bloom-filter union}
11: for all O in OVERLAYNEIGHBORS(N) do
12:     SENDTONODE(O, (N, H))
```

from being used to discover anycast group members, and BubbleStorm [13] from being used to discover different copies of a file. Quasar solves this problem using the novel notion of *negative information*.

Negative information is attached to each message that is used to cancel out gravity wells of previous members along the path, thereby averting loops. This is enabled by disseminating a unique, but aggregatable, node ID alongside the proactive dissemination of routes to groups. The routing algorithm uses the negative information, consisting of a set node IDs, to disable the gravity wells of these nodes on a per-message basis. Overall, using both proactive dissemination and negative information, Quasar creates a novel rendezvous-less routing primitive for group communication.

3 SYSTEM DESCRIPTION

In Quasar, each node in the overlay represents a user in the social-network. Nodes keep track of the group membership of their users, and disseminate this information in an aggregate form to their overlay neighbors. Groups and nodes are uniquely identified in the system by the 160-bit hash of their names. While group and node IDs may share the same ID space as a matter of implementation convenience, unlike DHTs, Quasar does not enforce a semantic mapping between group and node IDs; node IDs are not used to discover groups.

3.1 Join Protocol

When a user subscribes to a group, he notifies his overlay neighbors that, in turn, propagate the notification further into the network. In addition to the group ID of the subscription, the notification includes the unique node ID of the node. Subscriptions are aggregated using attenuated bloom filters [11] and deliveries are batched.

An attenuated bloom filter with depth parameter K is a collection of K identically-sized bloom filters where the n-th bloom filter holds information about the groups and node IDs n-hops away. Each node maintains a *home*-attenuated filter to disseminate to its neighbors. To illustrate, a node's 0-th level bloom filter contains objects at its own node, consisting of its node ID and its group memberships. Its 1-st level bloom filter contains all objects available at 1-hop neighbors and neighbor IDs, constructed from the union of the level-0 bloom filters of all its neighbors. Its 2-nd level bloom filter contains all objects and IDs 2-hops away, and so on. As higher level bloom filters are constructed through unions of lower level filters, the false positive rate, a function of the number of entries collected in the filter, increases at each level. Intuitively, higher level bloom filters have a broader but more diluted view of the network, while the lower level filters have a narrower but more precise view.

In practice, each node receives and maintains a depth-K attenuated bloom filter for each overlay neighbor. Soft-state with periodic refresh is used to update routes and reap stale routes.

Algorithm 1 describes the join protocol in pseudocode. The size of the attenuated bloom filter (B) and its depth (K) are system parameters; their effects on performance is explored later in this paper. The soft-state timeout (T) is selected to balance freshness and communication costs.

Fig. 1 illustrates the join process for multiple members in a group. In the figure, each funnel represents a gravity well created by a single node joining the group. Since subscription notifications include both the group ID and the node's unique ID, each funnel is actually a superposition of two separate gravity wells — one indexed by the group ID in the bloom filters, and the other indexed by the node ID. Messages routed based on the group ID can sink into any matching gravity well, however, node IDs in combination with negative information in the message, steer the message away from individual gravity wells.

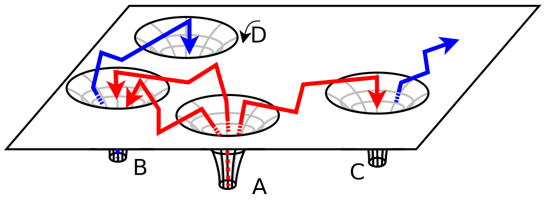

Figure 2: Probabilistic message routing to group members. A publishes a message using parallel random walks (shown in red). The message is not affected by A's gravity well due to negative information. Message encounters gravity wells of B and C, and is delivered to them. B and C re-publish the message using parallel walks (only one shown for clarity) after appending their respective node IDs to the negative information of the message.

3.2 Routing Protocol

Publishing a message to a group involves routing the message to multiple group members. The routing combines parallel random walks to find gravity wells, followed by directed walks into these wells. Each message in Quasar is addressed to a group ID, and is tagged with negative information consisting of the node ID of one or more publishers. At each hop, a node routing the message checks if the message destination matches the ID of a group it is subscribed to, and delivers it to the user if it does. Otherwise, the node checks the bloom filter of each neighbor, in ascending order from level-0 to level-K. If a match is found, then the message has encountered a gravity well matching the target group; however, the gravity well might be that of a publisher and may need to be negated by the negative information. Negative information becomes active if the group ID and a publisher ID in the message is found in the same level bloom filter of a neighbor node. If this is not the case, the message is forwarded to the neighbor who's bloom filter yielded a match. If the gravity well is negated or no neighbor yields a match, the message is routed to a random neighbor until the TTL expires.

The routing above delivers a message from a publisher to a nearby group member. Multiple copies of the messages are sent out in parallel in different directions to reach multiple group members. Newly delivered messages are re-published with the same branching factor to reach additional group members. Re-published messages include the ID of the original publisher along with IDs of all subsequent re-publishers of the message.

Algorithm 2 lists pseudo code for the Quasar routing algorithm, and Fig. 2 illustrates the routing through an example. In the figure, the original publisher A issues three copies of the message, which sink into the gravity wells of two nearby group members B and C. Each member issues additional copies, some of which are delivered to other members, while others are absorbed by the network through expired TTLs.

Algorithm 2 ROUTE(M)

```
Require: M = (G, PUB, TTL): Message to be routed
Require: G: Group ID of message
Require: PUB: List of publishing node IDs
Require: TTL: Time-to-live for message
Ensure: M is routed closer to group member when possible
 1: if ISDUPLICATE(M) then
 2:     return
 3: N ← GETLOCALNODEID()
 4: if G in SUBSCRIBEDGROUPS(N) then
 5:     DELIVER(M)
 6:     PUB ← PUB ∪ N {Re-publish message}
 7:     for all O in OVERLAYNEIGHBORS(N) do
 8:         SENDTONODE(O, M)
 9:     return
10: if DECREMENT(TTL) = 0 then
11:     return
12: for all L in 0...K do
13:     for all O in OVERLAYNEIGHBORS(N) do
14:         F ← GETATTENUATEDFILTERFOR(O)
15:         if F[L].CONTAINS(G) then
16:             NEGRT ← false
17:             for all P in PUB do
18:                 if F[L].CONTAINS(P) then
19:                     NEGRT ← true
20:             if not NEGRT then
21:                 return SENDTONODE(O, M)
22: O ← RANDOMOVERLAYNEIGHBOR(N)
23: SENDTONODE(O, M)
```

3.3 Impact of False Positives

The use of bloom filters to aggregate routing information potentially introduces false positives in routing table matches. A false positive in matching a target group to a gravity well represents an induced pull of gravity from an imaginary source. In such cases, the message will attempt to descend down the imaginary force vector, however, at the next hop, a match will not be found at a lower-level bloom filter. Consequently, message routing will fallback to a random walk. The result of a false positive here is a longer route.

A false positive in matching a node ID represents an imaginary publisher or re-publisher that can nullify the effects of a real gravity well. In this case, the message will not spiral down the gravity well, but instead take a random step. If the random neighbor does not have the same false positive, the message will sink into the real gravity well if it is still close enough to be affected by the well's gravity. However, with a very small probability, the message will be steered away if all the surrounding nodes of the gravity well exhibit the same false positives. Even in this case, a different copy of the message, with negative information for a different set of publish-

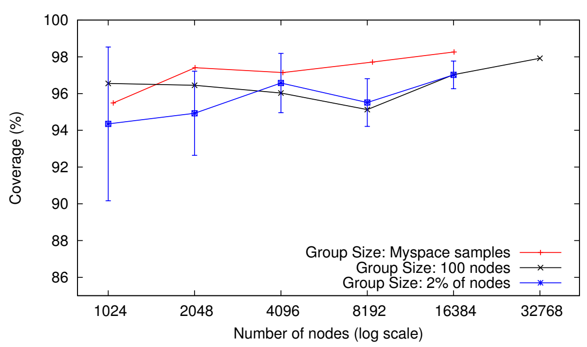

Figure 3: Quasar reaches a consistently high fraction of nodes independent of system size and group size

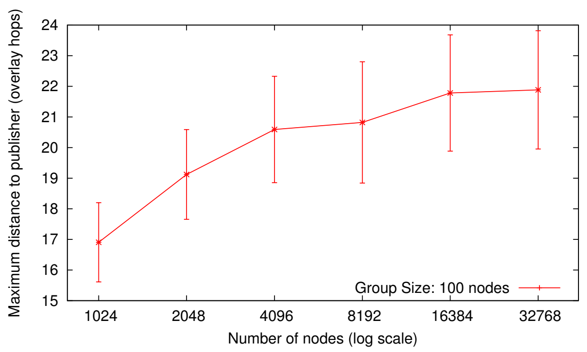

Figure 4: Quasar delivers messages quickly to group members

ers, can still sink into this real gravity well. The result of a false positive here, therefore, primarily affects performance but has a very small probability preventing message delivery to a member. A persistent failure to forward between two nodes is even less likely due to constant group membership changes and node churn.

3.4 Effect of Churn and Failed Nodes

While the effects of churn on the underlying overlay is out of the scope of this discussion, Quasar itself is highly resilient to churn. Neighbors engage in periodic exchange of bloom filters, which, coupled with the soft-state of gravity wells, provide automatic reaping of stale gravity wells over time.

During the state of transience when messages already in transit arrive at nodes that have no next-hop in its directed-route, the routing falls back to a random walk. To reduced the likelihood that a message gets induced back into the same stale gravity well, negative information for the node where the routing inconsistency was first detected is added to the message. This prevents the message from looping to the same node after traveling one or more random hops away.

4 EVALUATION

We evaluated Quasar through a large scale simulation parameterized with MySpace data. Nodes incurred a maximum storage overhead of 22 kB of bloom filters. The depth K of the attenuated bloom filter is set to 3 hops, and the size B of each bloom filter is set to 512 B.

MySpace data: We crawled MySpace to gather real data of group sizes in social networks. We crawled 10,930 MySpace profiles and discovered that friend lists, on average, contain 916 users of which only 37 are online at any given time. Plotting the group sizes on a log-log plot we noticed, unsurprisingly, two distinct Zipf-like trends in the data — one for normal users, and one for user communities focusing on celebrities, music bands, and TV personalities. We use this data to drive our simulations and choice of parameters.

The primary metric for Quasar's performance is "coverage". By design, Quasar lacks perfect information about group membership, instead dynamically discovering members during message routing. Coverage is defined to be the fraction of group members receiving a published message out of the total number of members in that group.

Fig. 3 shows that Quasar has very high coverage across a wide range of system sizes. The graph plots coverage as a function of the number of nodes in the system, which we vary from 1024 to 32768 in powers of two. The underlying network is a random graph with a constant network diameter. Each data point represents the average of 100 runs, with standard deviation error bars shown for one representative line.

In Fig. 3, the top line indicates that Quasar delivers messages to over 97% of group members independent of the size of the network, where group sizes were sampled from our MySpace dataset and ranged from between 2 to 1055. The two other lines show that system performance is consistently high as the system size grows for both a constant group size, typical for friends lists in social networks, and a group size proportional to system size, typical for growing online communities. Error bars for the latter confirm our intuition that Quasar's performance increases, which is manifested as lower variability, as the group sizes increase.

However, we expected coverage to steadily increase as we increased group size, but instead found that coverage peaked at around 97%. The reason for this, we discovered, was that as group size increases, some nodes are heavily obscured by numerous other group members, which collectively absorb messages from all directions. This suggests that combining Quasar with limited polling may increase coverage to 100%.

The second metric for measuring Quasar's performance is the upper bound on latency between publishing a message, and the last group member receiving that message. Fig. 4 plots the total number of overlay hops taken as a function of system size for a constant group size. As can be seen, the number of hops increases modestly as the system size changes by several orders of magnitude. The total number of hops observed is orders of magnitude smaller than the worst case random walk or dissemination tree, which would be the order of nodes and group members respectively, demonstrating the effectiveness of Quasar's proactive dissemination and negative routing.

The size of our bloom filters are chosen judiciously to reduce false positives based on our MySpace data. We detected only 0.5% false positives during routing, all of which Quasar recovered from at the subsequent hop. Extrapolating from our data, we believe that for a real world deployment the size of MySpace or Facebook, Quasar would require a mere 512 kB of storage at a node for comparable performance.

5 RELATED WORK

Application-level publish-subscribe systems has been a rich area of research in recent years. Much of the work has been focused on constructing efficient single-source multi-cast overlays [4, 1, 5] for specific workloads and environments. However, these systems are unsuitable for event dissemination in social networks as they can not be trivially made to support multiple groups, or multiple sources within a single group.

There have been many systems that provide the anycast primitive in an application-level overlay. Some [3] make use of a rendezvous node per group running on top of a structured overlay, while others make use of random-walk searches coupled with replication [6, 13]. Quasar is similar to the search-based systems, but disseminates aggregated routes rather than replicas, and introduces negative information, allowing it to be the basis for multicast.

Quasar is most closely related to systems that provide multi-cast for many simultaneous groups [14, 2, 10, 12, 9]. These systems rely on structured overlays and rendezvous nodes to keep track of and relay group information. Quasar provides a different point in the design space that requires significantly less complexity and maintenance than structured overlays and avoids hotspots from non-uniform popularity distribution of groups.

6 SUMMARY

In this paper, we described Quasar, a scalable, peer-to-peer publish-subscribe system that caters to the specific needs of social networks. It explores a new point in the publish-subscribe design space that uses a combination of proactive dissemination of aggregated routing vectors together with directed walks to provide an efficient anycast primitive for finding nearby group-members. This primitive, when coupled with negative information, a novel mechanism for dynamically negating routes, enables a scalable multicast to be built of top of the anycast primitive. The system shows much promise; evaluation on a MySpace trace shows that more than 95% of nodes are covered in the multicast, and the number of overlay hops from the source of the tree to the furthest leaf is modest and grows logarithmically to the system size.

References

[1] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu and Y. Minsky. Bimodal Multicast. *ACM ToCS* 17, 1999.

[2] M. Castro, P. Druschel, A. Kermarrec and A. Rowstron. SCRIBE: A Large-Scale and Decentralised Application-Level Multicast Infrastructure. *JSAC*, 2002.

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

[4] Y. Chu, S. Rao and H. Zhang. A Case for End System Multicast. In *SIGMETRICS*, Santa Clara, CA, June 2000.

[5] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek and J. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *OSDI*, San Diego, CA, Oct. 2000.

[6] D. Katabi and J. Wroclawski. A Framework for Global IP-Anycast (GIA). In *SIGCOMM*, Stockholm, Sweden, Aug. 2000.

[7] J. Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *STOC*, Portland, OR, May 2000.

[8] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells and B. Zhao. OceanStore: An Architecture for Global-Scale Persistent Storage. In *ASPLOS*, Cambridge, MA, Nov. 2000.

[9] A. Nandi, A. Ganjam, P. Druschel, E. Ng, I. Stoica, H. Zhang and B. Bhattacharjee. SAAR: A Shared Control Plane for Overlay Multicast. In *NSDI*, Cambridge, MA, Apr. 2007.

[10] V. Ramasubramanian, R. Peterson and E. G. Sirer. Corona: A High Performance Publish-Subscribe System for the World Wide Web. In *NSDI*, San Jose, CA, May 2006.

[11] S. Rhea and J. Kubiatowicz. Probabilistic Location and Routing. In *INFOCOM*, New York, NY, June 2002.

[12] D. Sandler, A. Mislove, A. Post and P. Druschel. FeedTree: Sharing Micronews with Peer-to-Peer Event Notification. In *IPTPS Workshop*, Ithaca, NY, Feb. 2005.

[13] W. Terpstra, J. Kangasharju, C. Leng and A. Buchmann. BubbleStorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search. In *SIGCOMM*, Kyoto, Japan, Aug. 2007.

[14] S. Zhuang, B. Zhao, A. Joseph, R. Katz and J. Kubiatowicz. Bayeux: An Architecture for Scalable and Fault-Tolerant Wide-Area Data Dissemination. In *NOSSDAV*, Port Jefferson, NY, June 2001.