A Measurement Study of a Publish Subscribe System

Hongzhou Liu   Emin Gün Sirer
Department of Computer Science, Cornell University
{liuhz,egs}@cs.cornell.edu

ABSTRACT
While publish-subscribe systems have attracted much research interest in the last decade, few established benchmarks have emerged and there has been little characterization of how they are used in practice. This paper examines RSS, a newly emerging, widely used publish-subscribe system for web micronews. Based on a trace study spanning 45 days at a medium-size academic department, and periodic polling of approximately 100,000 RSS feeds, we extract feed and client characteristics for RSS. We find that the popularity of RSS feeds follows a power law distribution. 16% of RSS feeds are updated hourly and 25% do not change at all during our polling period. 64% of all updates involve less than three lines in the XML document. We also find that RSS feed update sizes are proportional to feed size. Overall, an analysis of RSS, the first widely deployed publish-subscribe system, can help inform the design of next generation pub-sub systems.

1 Introduction

Publish-subscribe (*pub-sub*) systems are an emerging research area [3, 4, 11, 1], with applications that span information delivery, sensor monitoring, auction systems and air traffic control. Previous research in this area has focused on issues like system architecture, event routing and filtering algorithms, but has left a fundemental issue untackled, namely, what does the workload of a publish-subscribe system look like and how will clients use pub-sub systems in pratice?

This paper answers these questions by examining RSS, the first widely deployed pub-sub system. RSS is a simple topic-based information dissemination system for web micronews. Its architecture is quite simple: clients subscribe to a feed that they are interested in and poll the feed periodically to receive updates. The news items are encoded in XML, and displayed by a feed-reader or an RSS-integrated web browser on the client host. First introduced by Netscape as version 0.90 in 1997, RSS has become a popular format for syndicate web micronews. Most news media support RSS feeds, and other information, such as new articles on web sites and updates to weblogs are typically disseminated through RSS. Increased integration into browsers has recently made RSS accessible by mainstream users, who may not otherwise have access to feed-readers.

In this paper we use two different approaches to study the RSS system. First, we collect and analyze a 45-day trace from the Deparment of Computer Science at Cornell university. Our department network is used by 600 people, including graduate students, faculty and staff. The network is topologically separated from transient users, such as undergraduates in computer labs, who do not have dedicated computers and lack the ability to have long-running programs. Using this trace, we identify 158 different users that subscribe to 667 feeds in total. We examine the popularity of RSS feeds by the number of queries and subscribers, the number of feeds that a client subscribes to, how often a client polls the feeds and the activity time of the clients. Second, we survey RSS feed properties by actively polling 99,714 feeds collected from the feed directory syndic8.com at hourly intervals for 84 hours. We use these feed snapshots to distill feed properties such as the average feed size, update rate and update size information.

The results show that the popularity of RSS feeds follows a Zipf distribution. Most RSS feeds are small, ranging from 1KB to 10KB, with a median of 5781 bytes. 16% of RSS feeds update hourly and 25% do not change at all during the 84 hours period. Among those feeds that change, 64% of updates involve no more than three lines of XML. There is little correlation between feed size and update rates, though we find that the number of subscriptions per client follows a Zipf distribution. Finally, we find that, unlike web traffic, RSS traffic is "sticky" and constant; though RSS follows a very slight diurnal cycle similar to web traffic, it places an inelastic minimum load on web servers.

The paper is organized as follows. The next section presents background to pub-sub systems, RSS, and related work. We describe our measurement methodology in Section 3. We report the measurement results in Section 4 and conclude in Section 5.

2 Background and Related Work

In this section, we first provide some background on publish-subscribe systems and RSS, and then describe related work.

2.1 Publish-Subscribe Systems

Publish-subscribe is a distibuted computing paradigm that consists of three principal components: subscribers, publishers and an infrastructure for event delivery. Subscribers express their interest in an event, or a patterm of events. Publishers generate events. The infrastructure is responsible for matching events with the interests and sending them to the subscribers that registered the interest. Based on the way the subscribers specify their interest, pub-sub systems can be classified into two categories: subject-based and content-based. In subject-based pub-sub systems, subscribers specify their interest by subscribing to a *feed*, also known as *subject*, *channel*, *topic* or *group*. Each event produced by the publisher is labeled with a subject and sent to all the subscribers that subscribe to this subject. In other words, publishers and subscibers are connected together by a pre-defined subject. The major disadvantage of subject-based systems is their expressiveness: all the subjects are defined by the publisher and subscribers cannot further distinguish between events on a given subject. Content-based systems fix this problem. In such systems, subscribers specify their interest by event filters that are functions of event contents. Published events are matched against the filters and sent to the subscribers if they match the specified fitlters.

2.2 RSS

RSS is a Web content syndication system [10]. RSS concerns itself with propagating XML documents containing short description of web news. The XML documents are accessed via HTTP through URLs, and the URL for a particular XML document identifies that **RSS feed**. Programs called **RSS readers** check the contents of RSS feeds periodically and automatically on the user's behalf and display the returned results. Most feed readers poll RSS feeds once per hour by default. Newer versions of RSS support features like *TTL*, *SkipDay* and *SkipHour* that help RSS readers to decide when and how often to poll the feeds. Nevertheless, most RSS providers post a rate limit to prevent aggressive readers from overloading their servers.

The RSS system is a simple subject-based pub-sub system. Publishers publish their news by putting it into an RSS feed and provide the URL for the feed on their website. RSS users subscribe to a RSS feed by specifying its URL to the RSS reader. Thereafter, the RSS reader will poll the feed periodically and display the updates to the users if there are any.

Trace length	45 days
Number of clients	158
Number of feeds	667
Number of requests	28950

Table 1: Summary statistics for the user traces. Clients are identified by a secure crytographic hash of their IPs.

Polling period	84 hours
Number of feeds	99714
Number of snapshots	3682043
Bytes received	57GB

Table 2: Summary statistics for the active polling study.

2.3 Related work

Previous work on publish-subscribe systems has focused on the design and implementation of efficient event delivery systems. Isis [4], SIENA [3], Gryphon [8], TIBCO [9], Astrolabe [11] and Herald [2] are examples of publish-subscribe systems that have been proposed in the past. The Joint Battlespace Infosphere project [1] is a similar effort by the Air Force to provide a pub-sub based event notification and data repository system for very large scale deployment. FeedTree [6] is a recently proposed system designed to alleviate the load on RSS feed providers by cooperative polling using a distributed hash table for coordination. Previous work on characterizing user behavior on the web [12, 13] and in peer-to-peer systems [7, 5] is similar to our study in its flavor but not its domain. To the best of our knowledge, this study is the first characterization of feed properties and user behavior on a widely-deployed publish-subscribe system.

3 Measurement Methodology

The data in this paper are based on passively logging a 45-day user trace collected at the Department of Computer Science at Cornell University and on actively polling nearly 100,000 feeds every hour for 84 hours.

3.1 Passive Logging

We install software and hardware at the network border of our department to capture RSS traffic. The trace is collected over a 45 day period, spanning from March 22th to May 3rd, 2005. Our department is a medium-size academic organization with about 600 graduate students, faculty and staff. Table 1 provides a summary of the trace.

Our tracer software operates by capturing every TCP packet, reassembling full TCP flows, and logging the flows that contain an RSS request or response. For anonymity, we obfuscate client IP addresses using a one-way hash salted with a secret; this enables us to identify unique IP addresses without being able to map them

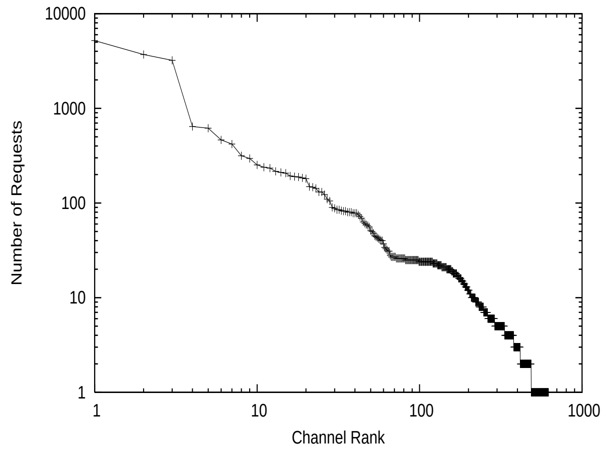

Figure 1: Feed rank by the number of queries received. RSS feed popularity follows a Zipf distribution.

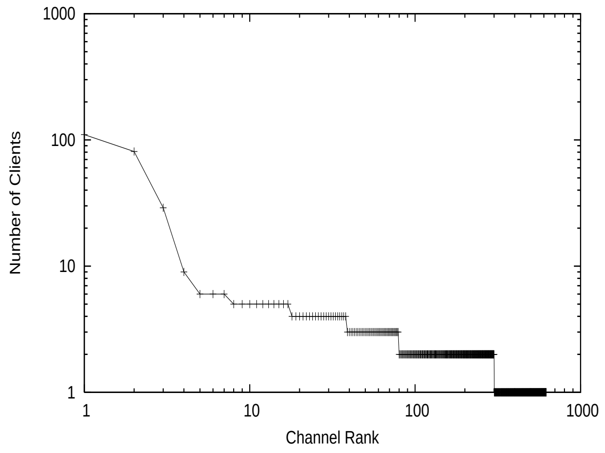

Figure 2: Channel rank by number of subscribers.

back onto hosts. Although DHCP is used in our department, the assignment of IP addresses is decided by the physical network port used and is therefore quite static. Laptop users that connect to public network ports may have different IPs over time but we estimate the number of laptop users is very low compared to users with fixed IPs. The software is run on a Dell dual processor 4650 workstation, which is able to keep up with packet capture at Gigabit line speed on the link from our department to the campus backbone (we make flow assembly non-performance critical by performing it offline on the captured packet stream).

3.2 Active Polling

We obtained a list of 99,714 RSS feeds from syndic8.com, a directory that acts as a vast repository of RSS feeds. We actively polled these feeds every hour for 84 hours, and recorded the results. While fetching the feeds, download timeout is set to 20 seconds and a request is retried 4 times if the response is not received within the timeout period. A successful download of the RSS contents gives a snapshot of the RSS feed at that time. An RSS feed can have 84 snapshots at most. We fetch 3682043 snapshots in total; that is, about 36.9 snapshots/feed. Failure to take a snapshot are mainly due to two reasons: the download fails because of network problems, or we exceed the polling rate limit of the RSS server. In order to calculate the feed update rate more precisely, we filter out all the feeds that have less than thirty snapshots, which yields 68266 feeds left. When calculating the feed update rate, we assume that a feed does not change at time t if the snapshot at that time is missing. This assumption will underestimate the feed update rate. The results of active polling are summarized in Table 2.

4 Measurement Results

In this section, we examine the characteristics of RSS feeds and clients. We examine properties that are inherent to RSS feeds, such as their size, update rate, and update size, as well as intrinsic properties of the client population, such as number of subscribed feeds, polling frequency, and hours of activity.

4.1 Feed Popularity

Figure 1 shows the popularity of RSS feeds ranked by the number of queries received. We can see that the popularity follows roughly a power law distribution with α equal to 1.24. The most popular feed (BBC news) receives 5704 requests while there is a long tail with many feeds that receive only a single request. Figure 2 plots the popularity of RSS feeds based on number of subscribers observed in the trace. The distribution of subscribers can also be characterized as a Zipf distribution. Though the log-log plot amplifies the divergence from Zipf on the left-hand side, the vast number of datapoints towards the right-side of the graph follow a Zipf distribution.

4.2 Feed Size

RSS feeds typically consist of web content encapsulated in XML format. Therefore, we expect the majority of RSS feeds have size close to most web objects. This is confirmed by Figure 3 that shows the distribution of feed size. As we can see, most (80%) RSS feeds are relatively small at less than 10KB. The minimum observed feed size is 356 bytes, median is 5.8KB, and the average is 10KB. While more than 99.9% of feeds are smaller than 100KB, we also see a long tail, not shown in the graph, with the largest feed at 876,836 bytes.

When a client detects an update of the RSS feed, it will fetch the whole feed regardless of the size of the update. Therefore, the bandwidth consumption under the current RSS system architecture is proportional the size of the

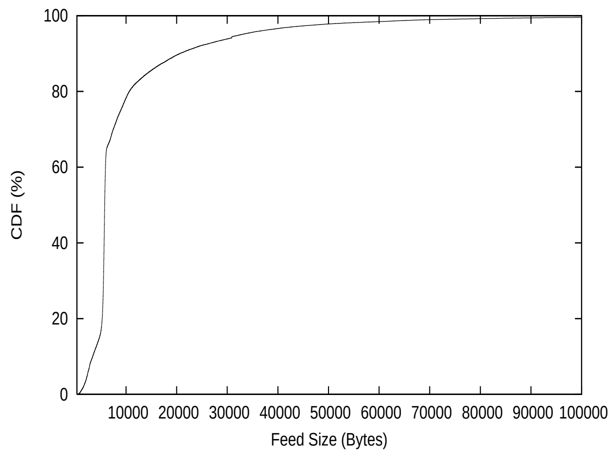

Figure 3: CDF of RSS feed size.

feed. This discourages content providers from supporting large feeds and biases towards small feed sizes.

4.3 Feed Update Rate

Feed updates are the main driving force behind the popularity of the RSS pub-sub system. In this subsection, we examine how often RSS feeds are updated. Figure 4 shows the distribution of intervals between the first successful snapshot and the next update. We can see that feeds are updated in two extremes: they either update very fast or very slowly. More than 50% feeds are updated in the first two hours and 30% of feeds are updated after eight hours.

Figure 5 shows the average update interval of RSS feeds. Update interval is calculated as $84/n$, where n is the number of updates observed within the polling period. This calculation underestimates the update interval as described earlier. Nevertheless, we still see over 40% of RSS feeds have average update interval no more than two hours. There are 25% of feeds that are not updated at all during the polling period and are omitted from this graph.

4.4 Feed Update Size

In this section, we examine how much RSS feeds change when they are updated. We quantify update sizes using the minimum edit distance ("diff") between two consecutive snapshots. Figure 6 shows the cumulative distribution of update sizes. 64% of all updates involve no more than two lines of changes. The average number of lines of that change is 16.7 and the maximum is 16542. The feed that changes most is hosted by a weather service website that provides weather forecast for many areas.

The major criticisms against RSS have centered around its scalability. The constant polling from clients pose a significant bandwidth challenge on RSS servers. There have been many proposals for reducing the band-

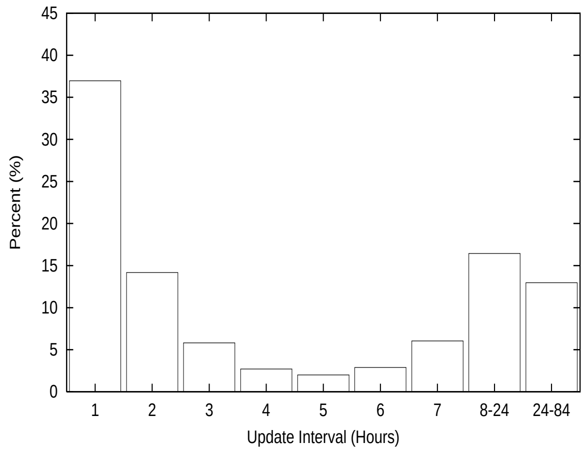

Figure 4: Update interval distribution for RSS feeds.

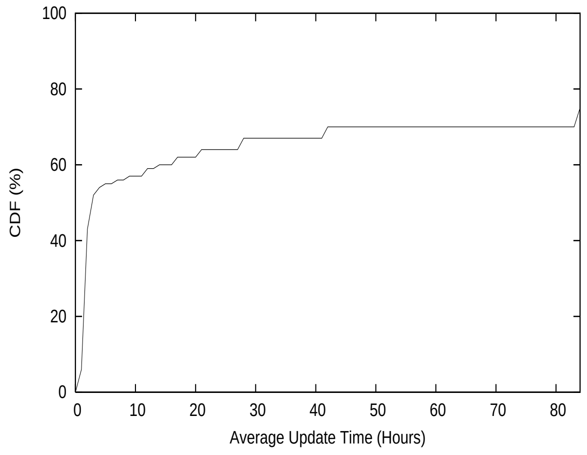

Figure 5: The average update time of RSS feeds.

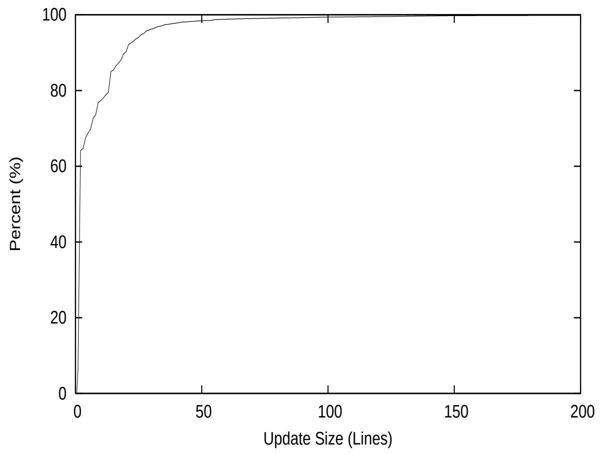

Figure 6: The number of lines that change when an RSS feed is updated.

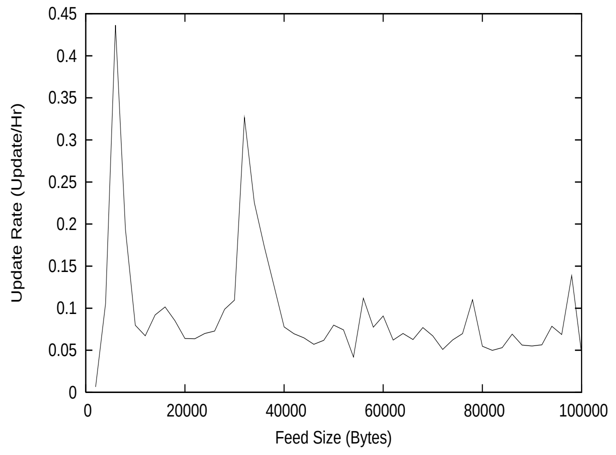

Figure 7: The correlation between feed size and update rate.

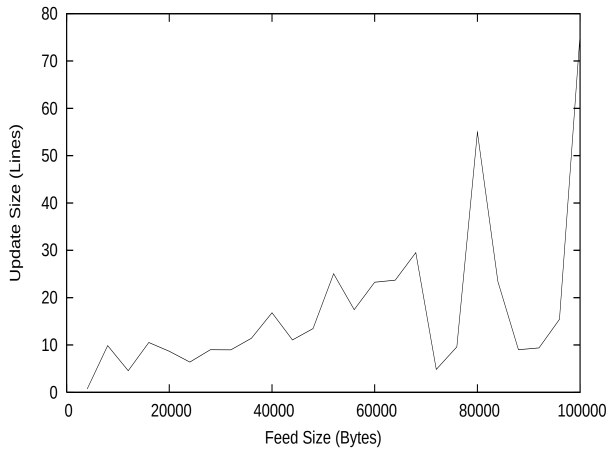

Figure 8: The correlation between feed size and update size.

width consumption. For instance, RSS 2.0 supports the *TTL*, *SkipDays* and *SkipHours* tags to advise the clients to choose an optimal polling rate and to skip periods when no updates are available, such as weekends. But a better solution is to send clients only the "delta," that is, the portion of data that actually changes. Our results indicate that this optimization can reduce bandwidth consumption significantly.

4.5 Feed Size and Update Relationship

In this subsection, we explore the correlation between feed size and update rates. First, we calculate the average number of updates as a function of feed size. The results are shown in Figure 7. Though the data indicates some peaks, there is no strong correlation between size and update rate. We suspect that the peaks are due to commonly used, frequently changing XML objects clustered around certain sizes. However, there is a correlation between feed size and update size, as can be seen in Figure 7. For most feeds, the average update size grows as feed size increases. The curve becomes irregular after feed size increases more than 60KB due to the small number of samples available.

4.6 Client Characteristics

We examine client characteristics in this subsection.

4.6.1 Overall Activity

Figure 9 summarizes the overall client activity over the 45 day trace period. We first examine the trace to determine if there is a change of system dynamics over time, specifically between busy daytime hours (8am to 8pm), and the relatively idle night time (8pm to 8am), as well as between weekdays and weekends. The periodic reductions in RSS traffic correspond to weekends. Unlike web traffic, RSS traffic does not exhibit a strong diurnal cycle, but we do observe more activity during daytime

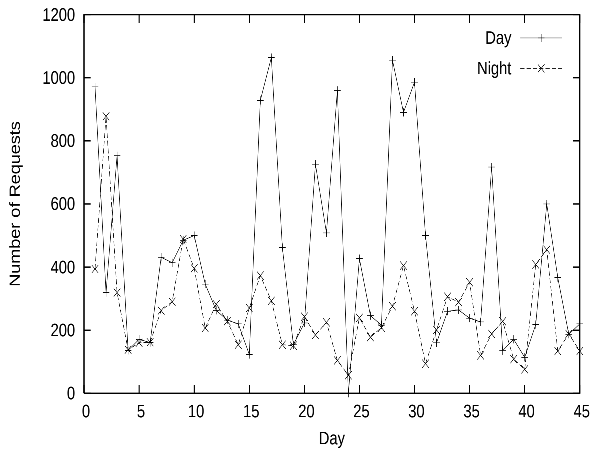

Figure 9: The overall activity of RSS clients over a 45 day period.

than night, at a ratio of 1.49 to 1. We also find that 42 clients (27%) poll RSS feeds only during the daytime, and that client activities decrease modestly on weekends. However, there is a significant baseline load generated by clients that are always on.

4.6.2 Number of Subscriptions

Figure 10 shows the ranking of the number of subscriptions per client. This distribution also follows a Zipf curve with α parameter around 1. While most clients subscribe to less than five channels, there are several clients that subscribe to more than 100 channels.

5 Conclusion

In this paper, we perform a measurement study of RSS feeds and clients based on results from passive tracing and active polling. We characterize the RSS feed popu-

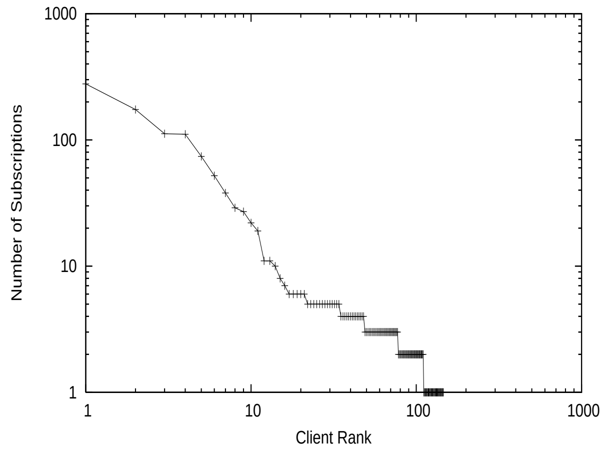

Figure 10: Number of channels subscribed by each client.

larity, size, update rate and update size, as well as client activity. Our results show that RSS feeds, like Web objects, follow a Zipf like popularity distribution and are usually small, under 10KB. Our results reveal the correlations between feed size and update characteristics. Feed update sizes are directly correlated with the size of the feeds. Our results about feed update size show a large space for improving bandwidth usage via delta encoding when updating RSS feeds. Publish-subscribe systems are a recently emerging, poorly characterized area. We hope our study can help to understand, design and evaluate future pub-sub systems.

References

[1] Air Force Research Laboratory (AFRL/IF) JBI Team. Joint Battlespace Infosphere. *http://www.rl.af.mil/programs/jbi/*, 2005.

[2] L. F. Cabrera, M. B. Jones, and M. Theimer. Herald: Achieving a Global Event Notification Service. In *Proc. of the Workshop on Hot Topics in Operating Systems*, Elmau, Germany, May 2001.

[3] A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and Evaluation of a Wide-area Event Notification Service. *ACM Transactions on Computer Systems*, 19(3):332–383, August 2001.

[4] B. Glade, R. Cooper, R. van Renesse, and K. Birman. Light-Weight Process Groups in the ISIS System. *Distributed Systems Engineering*, 1(1):29–36, September 1993.

[5] K. P. Gummadi, R. J. Dunn, S. Saroiu, S. D. Gribble, H. M. Levy, and J. Zahorjan. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. In *Proc. of the ACM Symposium on Operating Systems Principles*, Bolton Landing, NY, October 2003.

[6] D. Sandler, A. Mislove, A. Post, and P. Druschel. FeedTree: Sharing Web Micronews with Peer-to-Peer Event Notification. In *Proc. of International Workshop on Peer-to-Peer Systems*, Ithaca, NY, 2005.

[7] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proc. of Multimedia Computing and Networking*, San Jose, CA, January 2002.

[8] R. Strom, G. Banavar, T. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. Sturman, and M. Ward. Gryphon: An Information Flow Based Approach to Message Brokering. In *Proc. of International Symposium on Software Reliability Engineering*, 1998.

[9] TIBCO. TIBCO Publish-Subscribe, http://www.tibco.com/software/.

[10] UserLand. RSS 2.0 Specifications. *http://blogs.law.harvard.edu/tech/rss*, 2005.

[11] R. van Renesse, K. Birman, and W. Vogels. Astrolabe: A Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *ACM Transactions on Computer Systems*, 21(2):164–206, May 2003.

[12] A. Wolman, G. Voelker, N. Sharma, N. Cardwell, M. Brown, T. Landray, D. Pinnel, A. Karlin, and H. Levy. Organization-Based Analysis of Web-Object Sharing and Caching. In *Proc. of the USENIX Symposium on Internet Technologies and Systems*, Boulder, CO, October 1999.

[13] A. Wolman, G. M. Voelker, N. Sharma, N. Cardwell, A. Karlin, and H. M. Levy. On the Scale and Performance of Cooperative Web Proxy Caching. In *Proc. of the ACM Symposium on Operating Systems Principles*, Kiawah Island, SC, December 1999.