# Client Behavior and Feed Characteristics of RSS, a Publish-Subscribe System for Web Micronews

Hongzhou Liu Venugopalan Ramasubramanian Emin Gün Sirer
*Department of Computer Science, Cornell University, Ithaca, NY 14853*
{liuhz, ramasv, egs}@cs.cornell.edu

## Abstract

*While publish-subscribe systems have attracted much research interest since the last decade, few established benchmarks have emerged, and there has been little characterization of how publish-subscribe systems 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 characteristics of RSS content and usage. We find that RSS workload resembles the Web in content size and popularity; feeds are typically small (less than 10KB), albeit with a heavy tail, and feed popularity follows a power law distribution. The update rate of RSS feeds is widely distributed; 55% of RSS feeds are updated hourly, while 25% show no updates for several days. And, only small portions of RSS content typically change during an update; 64% of updates involve less than three lines of the RSS content. Overall, this paper presents an analysis of RSS, the first widely deployed publish-subscribe system, and provides insights for the design of next generation publish-subscribe systems.*

## 1 Introduction

Publish-subscribe or *pub-sub* systems [1, 5, 6, 8, 9, 12, 13, 15] are gaining wide acceptance with applications spanning information delivery, sensor monitoring, auction systems, and air traffic control. Previous research in this area has focused on aspects, such as system architecture, event notification, and content filtering algorithms, but has left a fundamental issue untackled, namely what does the workload of a pub-sub system look like and how do clients use pub-sub systems in practice?

This paper answers these questions by examining RSS, the first widely deployed pub-sub system, which is used for disseminating Web micronews. The architecture of RSS is quite simple: clients subscribe to a feed that they are interested in and poll the feed periodically to receive updates. RSS content is encoded in XML and displayed by a feed-reader or an RSS-integrated Web browser on the client host. Most news media support RSS feeds, and information such as announcements on Web sites and updates to weblogs, is typically disseminated through RSS. Integration into Web browsers has recently made RSS accessible to Internet users and greatly increased the popularity of RSS.

In this paper, we study the feed characteristics and client behavior in the RSS system using data collected through a combination of passive logging and active polling. First, we recorded a 45-day trace from the Department of Computer Science at Cornell University. The trace consists of 158 different RSS users, who subscribe to 667 feeds in total. We use this trace to examine the characteristics of RSS workload, such as the popularity of RSS feeds, and user behavior, including polling rate and subscription patterns. Second, we collected snapshots of RSS content by actively polling every hour 99,714 feeds listed in the feed directory *syndic8.com*. We use the feed snapshots to distill content properties, such as feed size and format, and to analyze updates in terms of update rate and amount of change.

Our study provides several insights, some expected and a few surprising, into the characteristics of the RSS system. First, we find that the RSS workload bears resemblance to the Web workload. The popularity of RSS feeds is heavy-tailed and follows a Zipf distribution similar to Web objects [3]. The typical sizes of RSS feeds are comparable to Web objects, although extremely large RSS documents are rare; most RSS feeds range from 1KB to 10KB, with a median of 5.8 KB, compared to a median of about 4 KB for the Web [7].

Second, RSS content changes more often than Web objects, albeit with a wide distribution of update rates. Our periodic snapshots show that 55% of RSS feeds update within an hour, while 25% do not change at all during 84 hours of polling. Even though RSS content may change rapidly, the update, surprisingly, affects only a tiny fraction of the content; 64% of updates involve no more than three lines of the XML. RSS clients can save a tremendous 93% of bandwidth by fetching the "delta"s instead of the entire feed during polling.

Finally, our study reveals interesting behaviors of RSS users. We find that over a third of the clients fetch feeds manually and do not use automated RSS tools that poll and check for updates periodically. Among the rest, over a half of clients poll feeds hourly, which is the default setting of

| Trace length | 45 days |
|---|---|
| Number of clients | 158 |
| Number of feeds | 667 |
| Number of requests | 61935 |

Table 1: **Summary of User Traces: Clients are identified by a secure cryptographic hash of their IPs.**

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

Table 2: **Summary of Active Polling: Feeds were polled in hourly intervals.**

most RSS readers. These behaviors indicate that enabling RSS content servers to provide feed specific polling rates to RSS readers is a more efficient way to customize RSS polling than expecting clients to configure their readers.

The rest of this paper is organized as follows: The next section provides some background on pub-sub systems and RSS. Section 3 describes our methods for studying the RSS system, and Section 4 details the results of our study. Finally, we discuss the implications of our study and conclude in Section 5.

## 2 Background and Related Work

Publish-subscribe systems have raised considerable interest in the research community over the years. In this section, we provide some background on pub-sub systems, a brief overview of RSS, and then summarize related research in this area.

**Publish-Subscribe Systems**

Publish-subscribe is a distributed 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 pattern of events. Publishers generate events. The infrastructure is responsible for matching events with the interests and sending them to the subscribers that registered the interests. Based on the way the subscribers specify their interest, pub-sub systems can be classified into two categories: topic-based and content-based. In topic-based pub-sub systems, subscribers specify their interest by subscribing to a *feed*, also known as *topic*, *channel*, *subject*, or *group*. Each event produced by the publisher is labeled with a topic and sent to all the subscribers that subscribed to this topic. In other words, publishers and subscribers are connected together by a predefined topic. The major disadvantage of topic-based systems is their expressiveness: all the topics are defined by the publisher and subscribers cannot further distinguish between events on a given topic. Content-based systems fix this problem. In such systems, subscribers specify their interest through event filters, which are functions of event contents. Published events are matched against the filters and sent to the subscribers if they match the specified filters.

### 2.1 RSS

RSS is a Web content syndication system [14] concerned with the propagation of XML documents containing short descriptions of Web news. The XML documents are accessed via HTTP through URLs, and the URL for a particular XML document identifies the *RSS feed*. Client applications 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 such as *TTL*, *SkipDay*, and *SkipHour*, which 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 topic-based pub-sub system. Publishers publish their news by putting it into an RSS feed and providing the URL for the feed on their website. RSS users subscribe to a RSS feed by specifying its URL to their RSS readers. Thereafter, the RSS readers will poll the feed periodically and display the updates to the users.

### 2.2 Related work

Previous work on pub-sub systems has focused on the design and implementation of efficient event delivery mechanisms. Isis [8], Linda spaces [5], T-space [9], SIENA [6], Gryphon [12], TIBCO [13], Astrolabe [15], and Herald [4] are examples of pub-sub systems 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 [11] and CorONA [10] are recently proposed systems designed to alleviate the load on RSS feed providers by cooperative polling using a distributed hash table for coordination. While a vast amount research material on pub-sub systems are available, this is the first measurement study of a widely-deployed pub-sub system.

## 3 Measurement Methodology

We investigate the characteristics of the RSS system from data collected through two techniques: passively logging a 45-day user activity at the Department of Computer Science, Cornell University and actively polling nearly 100,000 feeds every hour for 84 hours. The rest of this section describes how we gathered the RSS data.

**Passive Logging**

We built a software tool for tracing RSS traffic and installed it at the network border of our department. Our department is a medium-size academic organization with about 600 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 for long-running programs. We traced user activity over a 45 day period, spanning from 22 March to 3

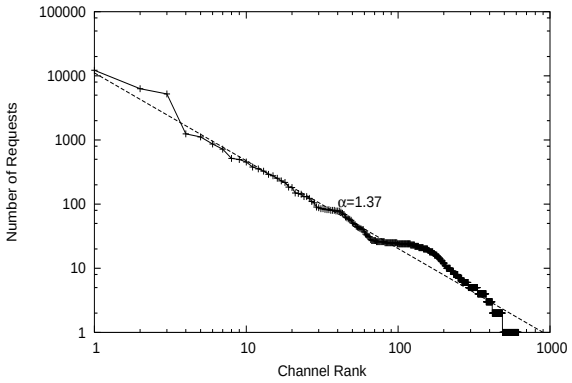


Figure 1: **Feeds Ranked by the Number of Requests: RSS popularity follows a Zipf distribution.**

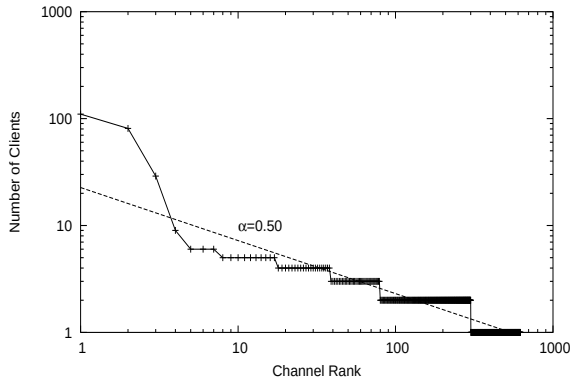


Figure 2: **Feeds Ranked by the Number of Subscribers: RSS popularity based on subscriptions also follows a Zipf distribution.**

May 2005, and recorded all RSS related traffic. 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 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 to be low compared to users with fixed IPs. The tracer tool ran on a Dell dual processor 4650 workstation, which was able to keep up with packet capture at Gigabit line speed on the link from our department to the campus backbone. We made flow assembly non-performance critical by performing it offline on the captured packet stream and observed no packet drops during the whole trace period.

**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 was set to 20 seconds and a request was retried 4 times if the response was not received within the timeout period. A successful download of the RSS content gives a snapshot of the RSS feed at that time. A download may fail due to high instantaneous load on the server, network congestion, or stringent polling limits imposed by servers. We fetched 3,682,043 snapshots in total; that is, about 36.9 snapshots per feed on average. The results of active polling are summarized in Table 2.

# 4 Survey Results

We report on three broad aspects of the RSS system using the trace data and periodic snapshots. First, we analyze the characteristics of RSS feeds, such as the popularity distribution, content size, format, and version of RSS used. Second, we investigate how RSS feeds are updated; in particular, we focus on the update intervals of RSS feeds, the amount of change involved in updates, and correlations between updates and feed size. Finally, we examine how clients use RSS by studying their polling behavior and subscription patterns. This section describes our findings in detail.

## 4.1 Feed Characteristics

We first present statistics on RSS workload and content. We compute the popularity of RSS feeds based on the user activity traces and derive content characteristics from the snapshots of RSS feeds. We measure popularity in two ways: based on the number of requests received for each RSS feed and based on the number of clients who subscribed to each RSS feed.

**Feed Popularity**

Figure 1 shows the popularity of RSS feeds ranked by the number of requests received. The popularity follows roughly a Zipf (power law) distribution with $\alpha$ parameter 1.37. The most popular feed (BBC news) receives 12,203 requests, while there is a long tail of 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 also follows a Zipf distribution ($\alpha$ = 0.5). The small number of clients in our trace makes the log-log plot diverge a little from the Zipf line. Overall, RSS workload has characteristics similar to Web workload, which is also known to follow heavy-tailed power-law distributions [3].

**Feed Format and Version**

We find that RSS is the widely-used format with more than 98% (97720 feeds) of the feeds in RSS; a small 2% (1994) of the feeds, however, use Atom [2], another XML based format for disseminating Web micronews. We further breakdown the RSS feeds according to their versions and show the results in Figure 3. Version 2.0 is the most popular format; more than 60% of RSS feeds published on *syndic8.com* are in this format. Version 0.91 and 1.0 count for about 17% each. Other versions(0.90, 0.93 and 0.94)

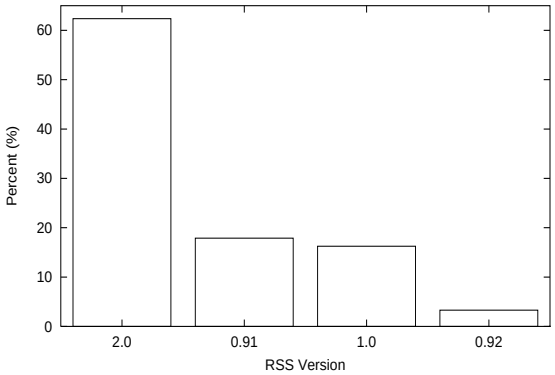


Figure 3: **Distribution of RSS Version. Version 2.0 is the most popular version of RSS.**

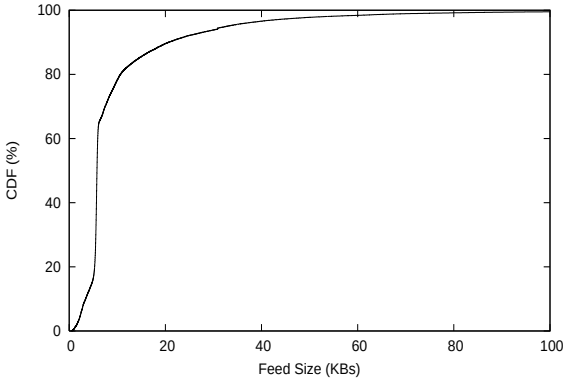


Figure 4: **CDF of Feed Size: RSS feeds are typically small (less than 10 KB) with a median of about 5.8 KB.**

are rare; they count for only 0.2% in total, and therefore are not shown in the figure.

### Feed Size

RSS feeds typically consist of Web content encapsulated in XML format. Therefore, we expect the majority of RSS feeds to have size close to most Web objects. This is confirmed by Figure 4, which plots the distribution of feed size. The feed size is calculated as the average of all the snapshots of the feed; the variance is very small for the feed snapshots. More than 80% of the 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, 99.9% of feeds are smaller than 100KB, the feed size distribution is heavy tailed with the largest feed at 876,836 bytes (not shown in the graph).

Extremely large RSS feeds, however, are rare, unlike some Web objects that can be of several megabytes or more. The concise nature of RSS feeds is expected because RSS is meant for the quick dissemination of news updates, often only carrying links to the more elaborate news articles. Moreover, the current architecture of RSS, where clients need to fetch the whole feed for checking updates, poses a high bandwidth load on content servers. This discourages content providers from supporting large feeds and biases towards small feed sizes.

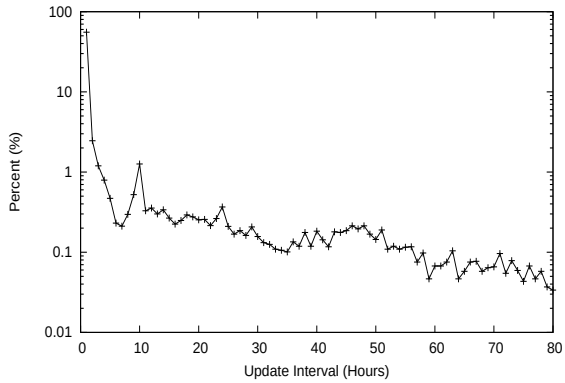


Figure 5: **Distribution of First Update Intervals: 55% of feeds get updated in an hour.**

## 4.2 Update Characteristics

Updates are the main driving force of the RSS pub-sub system. We examine the nature of RSS updates using the series of hourly snapshots gathered through active polling. We ensure that missing snapshots do not affect the calculations of update interval by only counting the intervals between *valid update*s; an update is valid only if there is a valid snapshot preceding the update, and that preceding snapshot matches the last recorded update. In order to calculate the update characteristics accurately, we filtered out all the feeds that have less than thirty snapshots leaving 68,266 feeds.

### Update Rate

Figure 5 shows the distribution of update intervals of the first valid update. We see that feed update rates fall in two extremes: they either update very frequently or very rarely. More than 55% of feeds are updated in the first hour, while 25% of feeds did not see any updates during the entire polling period. This result suggests that RSS readers should use different polling periods for different feeds. However, some RSS readers, e.g., Thunderbird 1.0, do not support this feature currently.

Figure 6 shows the average update interval of RSS feeds, calculated by averaging the valid update intervals measured for each feed. We see that over 57% of the RSS feeds have an average update interval under two hours. Since we gathered snapshots by the hour, our data do not show updates that happen within an hour. Nevertheless, we find that RSS feeds often change at a rapid rate and RSS readers need to poll aggressively in order to detect updates quickly.

### Update Size

We quantify update sizes using the minimum edit distance ("diff") between two consecutive snapshots. Figure 7 shows the cumulative distribution of update sizes. 64% of all updates involve no more than two lines of changes. The average change in the number of lines is 16.7 (6.8% of feed size) and the maximum is 16,542. The feed that changes most is hosted by a weather service website that provides weather forecast for many areas.

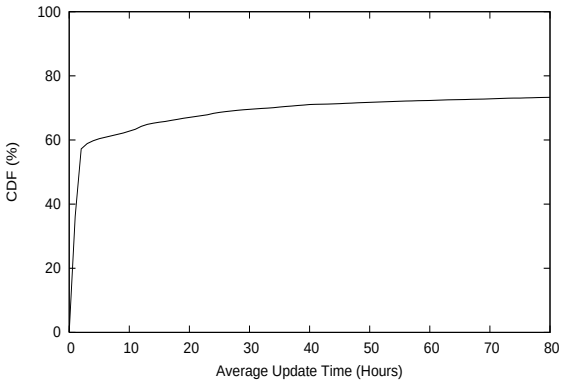


Figure 6: **Average Update Time: 57% of feeds have average update interval of less than two hours, while 25% of feeds do not change for more than three days.**

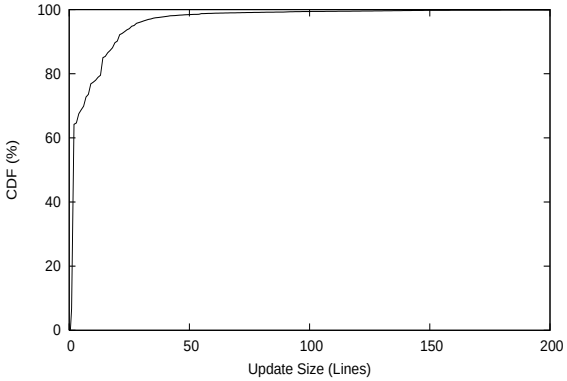


Figure 7: **Number of Changed Lines in Updates: 64% of updates involve no more than two lines of change.**

The major criticism against RSS has centered around its scalability. The constant polling by clients poses a significant bandwidth challenge on RSS servers. There have been many proposals for reducing the bandwidth 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 measurement shows that the feed updates only 6.8% of its content on average, which suggests that this optimization can reduce bandwidth consumption by as much as 93.2%.

### Correlations between Feed Size and Updates

We explore the correlation between feed size and update rates and sizes. Figure 8 shows the average number of updates as a function of feed size. 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 9. For most feeds, the average update size grows as feed size increases. For feeds smaller than 68KB (about 99% of the total), the correlation coefficient is 0.89. The curve becomes irregular

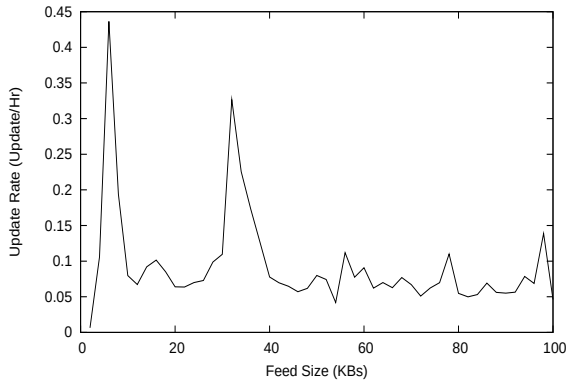


Figure 8: **Correlation Between Feed Size and Update Rate: There is no noticeable correlation between feed size and update rate.**

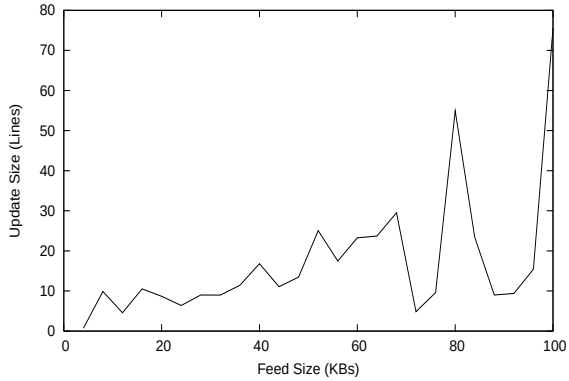


Figure 9: **Correlation between Feed Size and Update Size: The amount of change during an update increases with the feed size.**

after feed size increases more than 68KB due to the small number of samples available.

## 4.3 Client Behavior

Finally, we analyze how clients use the RSS system from the user activity trace we collected.

### Polling Frequency

We divide the clients into two categories, namely *auto and manual*, according to their polling behavior. Auto clients poll feeds at a fixed rate, usually by running RSS readers in the background, while manual clients use RSS in the same way as they browse the Web, that is, launch RSS readers when they really want to read the news, and close the program after reading it. We consider clients who poll a feed for less than 3 times a day or with irregular polling intervals as manual clients. We find that 36% of clients in our department fall in this category. For auto clients, who poll at periodic intervals, we show the polling rate in Figure 10. 58% of them poll feeds hourly, suggesting that most users are "lazy", and do not change the default setting of their RSS readers. A small number of aggressive clients poll as often as every ten minutes.

### Number of Subscriptions

Figure 11 shows the number of feeds subscribed by each client in sorted order. This distribution also follows a Zipf distribution with $\alpha$ parameter around 1.13. While most

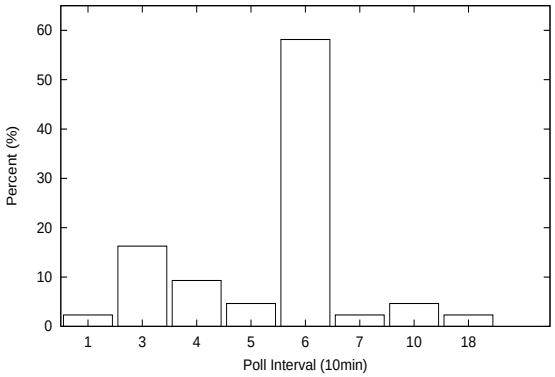


Figure 10: **Polling Rate of Clients: About 58% of clients use the default setting of one hour as the polling period.**

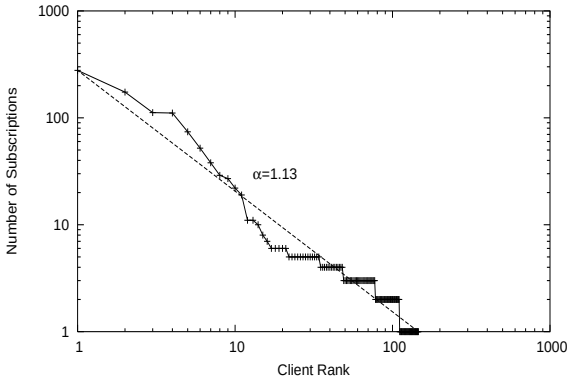


Figure 11: **Number of Subscriptions made by Clients: The number of channels subscribed by clients follows a Zipf distribution.**

clients subscribe to less than five feeds, there are several clients that subscribe to more than 100 feeds.

## 5 Discussions and Conclusions

This paper presents a measurement study of RSS, a pub-sub system for disseminating Web micronews. It provides insights about how a pub-sub system is utilized in practice and what issues need to be addressed while designing pub-sub systems.

The main focus of our study is to analyze how feeds are updated, a fundamental aspect of pub-sub systems. This study shows that update rates of RSS feeds are distributed in extremes; a majority of feeds (55%) update every hour, while many feeds (25%) do not change for days together. Hence, significant bandwidth savings can be obtained by using the optimal polling period for each feed instead of a single common polling rate for all feeds. End users of RSS, however, cannot be relied on to set the optimal polling rate, as this study shows that clients predominantly do not change the default settings of RSS readers. A better solution is for content providers to indicate when and at what rate to poll a particular feed. The version 2.0 of RSS already provides support for customized polling, although many readers are yet to support this feature.

Much of the bandwidth in RSS goes towards refetching feeds in order to check for updates because the current RSS architecture does not employ asynchronous notifications. This study indicates that delta encoding is a major opportunity for improving bandwidth usage in RSS, as updates are often made only to a tiny portion of the content (about 7% of the feed on average). Moreover, clients subscribed to the same feed poll the content servers independently, imposing a high load on the servers of popular feeds. Recently proposed systems [11, 10] use peer-to-peer overlays for cooperative polling to alleviate load on the servers and to provide faster updates. Such systems capable of asynchronous update notifications seem to be a step in the right direction.

Overall, this is the first study of a widely deployed pub-sub system performed during the early days of RSS. We hope this study will help to understand, design, and evaluate future pub-sub systems, and more studies with greater depth will emerge as the popularity of RSS increases.

## References

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

[2] Atom Enabled. Atom Syndication Format. *http://www.atomenabled.org/developers/syndication*.

[3] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web Caching and Zipf-like Distributions: Evidence and Implications. In *Proc. of IEEE International Conference on Computer Communications*, New York, NY, Mar. 1999.

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

[5] N. Carriero and D. Gelernter. Linda in Context. *Communications of the ACM*, 32(4):444–458, Apr. 1989.

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

[7] F. Douglis, A. Feldman, B. Krishnamurthy, and J. Mogul. Rate of Change and Other Metrics: a Live Study of the World Wide Web. In *Proc. of USENIX Symposium on Internet Technologies and Systems*, Monterey, CA, Dec. 1997.

[8] 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, Sept. 1993.

[9] IBM. TSpaces - Computer Science Research at Almaden. *http://www.almaden.ibm.com/cs/TSpaces/*.

[10] V. Ramasubramanian, R. N. Murthy, and E. G. Sirer. Corona: A High Performance Publish-Subscribe System for Web Micronews. *http://www.cs.cornell.edu/people/egs/beehive/corona*.

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

[12] 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*, Paderborn, Germany, Nov. 1998.

[13] TIBCO. TIBCO Publish-Subscribe. *http://www.tibco.com/software/*.

[14] UserLand. RSS 2.0 Specifi cations. *http://blogs.law.harvard.edu/tech/rss*, 2005.

[15] 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.