Sharing Web micronews with peer-to-peer event notification

Dan Sandler Alan Mislove Ansley Post Peter Druschel

Department of Computer Science
Rice University, Houston (TX)
{dsandler,amislove,abpost,druschel}@cs.rice.edu

Abstract

Syndication of micronews*, frequently-updated content on the Web, is currently accomplished with RSS feeds and client applications which poll those feeds. However, providers of RSS content have recently become concerned about the escalating bandwidth usage of RSS readers. Current efforts to address this problem by optimizing the polling behavior of clients sacrifice timeliness without fundamentally improving the scalability of the system. In this paper, we propose a micronews distribution system,* SCRIBE-RSS*, which uses peer-to-peer overlay networks to distribute efficiently RSS entries to subscribers as soon as they are available. Peers in the network share the bandwidth costs, which reduces the load on the provider, and updated content is available to clients immediately.*

1 Introduction

In the early days of the Web, static HTML pages predominated; a handful of news-oriented Web sites of broad appeal updated their content once or twice a day. Users were by and large able to get all the news they needed by surfing to each site individually and pressing `Reload`. However, the Web today has experienced an explosion of *micronews:* highly focused chunks of content, appearing frequently and irregularly, scattered across scores of sites. The difference between a news site of 1994 and a weblog of 2004 is its flow: the sheer volume of timely information available from a modern Web site means that an interested user must return not just daily, but a dozen times daily, to get all the latest updates.

This surge of content has spurred the adoption of RSS, which marshals micronews into a common, convenient format. Instead of downloading entire web pages, clients download an RSS "feed" containing a list of recently posted articles. However, RSS specifies a polling-based retrieval architecture, and the scalability of that mechanism is now being tested. There is growing concern in the RSS community over these scalability issues and their impact on bandwidth usage, and providers of popular RSS feeds have begun to abbreviate or eliminate their feeds to reduce the bandwidth stress of polling clients.

The current RSS distribution architecture, in which all clients periodically poll a central server, has bandwidth requirements which scale linearly with the number of subscribers. We believe that this architecture has little hope of sustaining the phenomenal growth of RSS [10], and that a distributed approach is needed. The properties of peer-to-peer (p2p) overlays are a natural fit for this problem domain: p2p multicast systems scale logarithmically and should support millions of participating nodes. Therefore, we propose SCRIBE-RSS, a new architecture for distributing RSS feeds that shares costs among all participants. By using p2p event notification to distribute micronews, we can reduce dramatically the load placed on publishers, while at the same time delivering even more timely service to clients than is currently possible. We go on to show how SCRIBE-RSS can be deployed incrementally and demonstrate how to secure SCRIBE-RSS in an open-membership environment.

The remainder of this paper is organized as follows. Section 2 provides background on RSS and the RSS bandwidth problem. Section 3 discusses related work to improve RSS, and Section 4 presents the design of SCRIBE-RSS. Section 5 concludes.

2 Background

2.1 RSS

RSS[1] refers to a family of related XML document formats for encapsulating and summarizing timely Web content. Such documents (and those written in the Atom syndication format [1], a recent entry in the specification

[1] There is some disagreement [4] over the exact expansion of this acronym. When Netscape first specified version 0.9 of RSS [17], it did so under the name "RDF Site Summary;" the acronym has since been taken to stand for "Rich Site Summary" or "Really Simple Syndication." The subtleties of the many debates over format versions, nomeclature, and ideology are omitted here.

fray) are called *feeds*. A Web site makes its updates available to RSS client software (variously termed "readers" and "aggregators") by making a feed available to HTTP clients alongside its conventional HTML content. Because RSS feeds are designed for machines instead of people, client applications can organize, reformat, and present the latest content of a Web site—or many sites at once—for quick perusal by the user. The URL pointing to this feed is advertised on the main Web site.

By asking her RSS reader to *subscribe* to the URL of an RSS feed, a user instructs the application to begin fetching that URL at regular intervals. When it is retrieved, its XML payload is interpreted as a list of RSS *items* by the application. Items may be composed of just a headline, an article summary, or a complete story in HTML; each entry must have a unique ID, and is frequently accompanied by a permanent URL ("permalink") to a Web version of that entry. To the user, each item typically appears in a chronologically-sorted list; in this way, RSS client applications have become, for many users, a new kind of email program, every bit as indispensable as the original. An RSS aggregator is like an inbox for the entire Internet.

2.2 RSS Bandwidth

Just as major news outlets have begun to discover RSS and to expose their audiences to this burgeoning technology [10, 11, 13], the RSS technical community is abuzz with weaknesses exposed by its runaway adoption. Chief among these is the so-called "RSS bandwidth problem." Essentially, Web servers which make RSS feeds available tend to observe substantially greater traffic loads as a result, out of proportion to any observable interactive visitor trend. Consequently, some sites have implemented self-defense mechanisms (*e.g.* smaller RSS feed sizes, or enforced limits on access) in an attempt to address the problem [12]. This situation is most likely the effect of many behaviors working in concert:

Polling. For each feed to which a user is subscribed, an RSS application must issue repeated HTTP requests for that feed according to some set schedule. Sites which offer RSS feeds must satisfy one request for every user, many times a day, even if there is no new content.

Superfluity. The RSS data format is essentially static; all entries are returned every time the feed is polled. By convention, feeds are limited to some N most recent entries, but those N entries are emitted for every request, regardless of which of them may be "new" to a client. While this bandwidth problem could be helped by introducing a diff-based polling scheme, all such requests must be processed by the RSS provider which adds more processing load.

Stickiness. Once a user subscribes to an RSS feed, she is likely to retain that subscription for a very long time, so this polling traffic can be counted on for the foreseeable future. If a previously-obscure Web site becomes popular for a day, perhaps by being linked to from popular Web sites, its browsing traffic will spike and then drop off over time. However, if that site offers an RSS feed, users may decide to subscribe; in this case, the drop in direct Web browsing is replaced by a steady, unending load of RSS client fetches. Such a Web site might be popular for a day, but it may satisfy a crowd forever. [23, 21]

Twenty-four-hour traffic. RSS client applications are commonly running on desktop computers at all hours, even when a user is not present; the diurnal pattern of interactive Web browsing does not apply. While the global nature of Web users may generate "rolling" 24-hour traffic, global use of RSS readers generates persistent 24-hour traffic from all over the Earth.

It is easy to see how a website may suffer for publishing RSS feeds. The most popular feed on Bloglines[2] is Slashdot[3], which has about 17,700 subscribers as of this writing. If each of those subscribers were using personal aggregation software (desktop clients), Slashdot's headlines-only RSS feed (about 2 kilobytes for a day's worth of entries, and typically polled half-hourly) would be transferred 850,000 times a day, for a total of 1.7 GB of data daily. *The New York Times* recently introduced a suite of RSS feeds[4] for its headlines; the front page alone claims 7,800 subscribers, but the sum of subscribers to all its feeds comes to 24,000. Feeds from the *Times* tend to be around 3 KB, or 3.5 GB of data per day with 30-minute polling. For websites wishing to provide their RSS readers with deeper content, the problem is worse still. Boing Boing[5], a popular weblog, chooses to publish complete HTML stories in RSS and Atom; 11,500 subscribers might receive 40 KB for each RSS request. To provide this service, Boing Boing must be able to accommodate 22 GB/day of RSS traffic alone. If the BBC News Web site is truly "updated every minute of every

[2] Bloglines (http://bloglines.com), a popular Web-based RSS reading application, offers subscription figures for the feeds it aggregates. We will use these figures (as of late October 2004) as a very crude approximation of reasonable RSS readership. Though Bloglines certainly polls RSS feeds only once for its thousands of subscribers, anecdotal evidence suggests that traditional desktop RSS client usage outweighs Web-based client usage, so we can regard these figures as a lower bound on overall RSS polling load.

[3] http://slashdot.org

[4] http://nytimes.com/rss

[5] http://boingboing.net

day,"[6] its RSS subscribers (18,000 to its various feeds on Bloglines) are unable to take advantage of it: the bandwidth demands of those subscribers polling every minute would be insatiable.

2.3 Scribe

Scribe [8] is a scalable group communication system built on top of a peer-to-peer overlay such as Pastry. Each Scribe group has a 160 bit *groupId* which serves as the address of the group. The nodes subscribed to each group form a multicast tree, consisting of the union of Pastry routes from all group members to the node with nodeId numerically closest to the groupId. Since membership maintenance is distributed in the tree, Scribe can handle highly dynamic groups.

Scribe achieves its scaling behavior because it reuses a single overlay to manage many groups, thus amortizing the overlay maintenance operations over many dynamic groups. The use of PNS overlay construction [6] yields efficient locality based group trees. These two facts allow membership changes to be very cheap, less than $\log n$ local messages.

3 Related Work

3.1 Improving the polling process

Several proposals have been submitted to ease the pain of RSS on webmasters. Many of these are described in detail in the RSS Feed State HOWTO [16]; examples include avoiding transmission of the feed content if it hasn't changed since the client's last request, GZip compression of feed data, and clever ways to shape the timetable by which clients may poll the RSS feed.

Unfortunately, because the schedule of micronews is essentially unpredictable, it is fundamentally impossible for clients to know *when* polling is necessary. Werner Vogels puts it succinctly: Polling Does Not Scale [22].

3.2 Outsourced aggregation

Several online RSS service providers (essentially, Web-based RSS readers) have proposed alternative solutions [2, 3]. In these "outsourced aggregation" scenarios, a centralized service provides a remote procedure interface which end-user applications may be built upon (or refactored to use). Such an application would store all its state—the set of subscribed feeds, the set of "old" and "new" entries—on the central server. It would then poll only this server to receive all updated data. The central RSS aggregation service would take responsibility for polling the authoritative RSS feeds in the wider Internet.

This certainly addresses the bandwidth problem, in a way: A web site owner will certainly service fewer RSS requests as end users start polling the central service instead. The operators of these central services will definitely have bandwidth issues of their own: they will now be at the center of all RSS traffic.

There is a far more insidious danger inherent in this approach, however: **a central point of control, failure, and censorship** has now been established for all participating users. A central RSS aggregation service may: *(i)* experience **unavailability or outright failure,** rendering users unable to use their RSS readers, *(ii)* elect to **discontinue or change the terms of its service** at any time, or *(iii)* silently **modify, omit, or augment RSS data** without the user's knowledge or consent.

Modification of RSS data by the central aggregator may come in the form of optimized or normalized RSS formatting (a useful feature, since syndication formats found in the wild are frequently incompatible [19]), but might take more dangerous forms as well: it may modify or corrupt the entries in a feed, or it may add advertising or other supplemental yet non-indigenous content to those feeds.

In summary, a third party may not be a *reliable* or *trustworthy* entity, and so it cannot be guaranteed to proxy micronews for client applications. For these reasons, centralized RSS aggregation is most likely not a viable long-term solution.

4 Scribe-RSS

4.1 Group communication with overlay networks

The obvious alternative to polling for data is to distribute that data, as it becomes available, to lists of subscribers. This approach may be adequate for small subscription lists (for example, e-mail lists), but it will not scale to accommodate the growing subscription demands of Web site syndication. Furthermore, while such an approach may reduce the overall bandwidth usage of RSS (by reducing unnecessary fetches), it does nothing to alleviate the per-update stress on network links close to the source.

To address these problems, we look to peer-to-peer overlay networks, which offer a compelling platform for self-organizing subscription systems. Several overlay-based group communication systems, including Scribe [8], offer distributed management of group membership and efficient routing of subscription events to

[6] As advertised on http://news.bbc.co.uk.

interested parties in the overlay.

We propose an RSS distribution system based on peer-to-peer subscription technologies called SCRIBE-RSS. In SCRIBE-RSS, timely Web content is distributed to interested parties via a Scribe-like subscription architecture. Although we chose to base this design on Scribe, there is no reason it could not be deployed on any group communication system which provides similar performance characteristics. In such a system, content may be distributed as soon as it becomes available; interested parties receive these information bursts immediately, without polling the source or stressing network links close to the source.

4.2 Architecture

When SCRIBE-RSS publishing software wishes to deliver an update to subscribers, the following steps are taken (in addition to refreshing a conventional RSS feed URL):

- **A complete RSS document is created** to contain one or more pieces of timely micronews.
- The RSS data is then **signed with the publisher's private key.** This is essential to establishing the authenticity of each published item.
- The signed RSS document is **multicast in the overlay** to those peers who have subscribed to a Scribe group whose topic is (a hash of) the feed's **globally unique ID**, trivially defined to be the canonical URL of the advertised RSS feed.
- Peers receiving the message verify its signature, parse the RSS data, and add it to the local RSS application state as if it were a conventional, polled RSS feed. **The user can be notified immediately** of the new entries.

SCRIBE-RSS aware client applications should be able to examine conventional RSS feed data to discover if updates to that feed will be published through Scribe. To do this, SCRIBE-RSS metadata should be added to the RSS document structure to signal that it is available for subscription in the overlay. This metadata (in the form of a `<subscribe>` tag within the main `<feed>` element) can carry additional relevant payload, such as the identity and cryptographic public key of the publisher. In this way, a SCRIBE-RSS application bootstraps the subscription process with a one-time HTTP request of the conventional feed. All future updates are distributed through incremental RSS items multicast in Scribe.

It is desirable for all publishers to cryptographically sign their published RSS data, so that clients may be able to trust the Scribe events they receive.[7] The public key can be represented by a `<publicKey>` child of the `<subscribe>` tag; it may be encoded inline or may be referenced with a URI and fingerprint. Clients should use the publisher's key verify the contents of any received RSS items to detect tampering.

Additionally, SCRIBE-RSS clients may be offline for periods of time and may miss RSS updates. Clients coming online should "catch up" by examining the HTTP-based RSS feed for previously-unseen items during bootstrapping. Every item in the feed is assigned a monotonically increasing sequence number so missed members of the feed can be detected. If membership changes and an item in the feed is missed by a client (as detected by a gap in the sequence numbers), the client may query its parent to recover that item. In order to satisfy such a request, each member of the system will keep a small fixed buffer of size n with the last n items in the feed. As a fallback, missing items may be recovered by querying the HTTP-based RSS feed as in the bootstrapping phase. This is expected to be rare as membership changes are usually detected and repaired quickly via the Scribe heartbeat mechanism.

4.3 Adoption and deployment

The proliferation of conventional RSS has depended largely on the availability of quality tools to generate RSS data; SCRIBE-RSS will be no different. Developers have several opportunities to provide support for this system. We break down the deployment scenarios into those which support SCRIBE-RSS fully, and those which serve as "adapters" to ease transition for legacy RSS systems.

4.3.1 Full SCRIBE-RSS support

Publishers. Web content management systems (*e.g.* weblog publishing packages, traditional workflow-based CMS software) join the overlay by becoming long-lived Pastry nodes with Scribe support. When new content is posted, the publishing software automatically creates a new SCRIBE-RSS multicast message and routes it to all online subscribers.

Readers. RSS-reading applications join the overlay as well. By doing so, they become part of the global Scribe service, distributing the network and processing loads of RSS event forwarding. The user interface for an RSS client should remain unchanged;

[7] Even though the general benefits of signed content are independent of the SCRIBE-RSS architecture, we believe our design offers both an excellent opportunity and a compelling need to introduce signed RSS.

the user subscribes to RSS feeds as she would do ordinarily, and the software takes care of detecting and bootstrapping a SCRIBE-RSS subscription if it is available. New RSS items are made available to users as soon as the corresponding Scribe events are received by the application.

4.3.2 Incremental SCRIBE-RSS support

Publishers. Legacy publishing software which currently emits valid RSS can be adapted to SCRIBE-RSS with a "republishing" engine running on (or near) the Web server. This process would poll the legacy RSS feed on an aggressive schedule, sifting out new content and distributing it via Scribe. Such a republishing tool might even be operated by a third party, in case the owner is slow to deploy SCRIBE-RSS. (This is already a common emergent behavior of the RSS community; several Web sites currently "scrape" the HTML of popular sites and redistribute that content in RSS format. It is up to a user to decide whether or not to trust this third-party proxy feed.)

Readers. Until RSS applications support SCRIBE-RSS natively, users can still contribute to the RSS bandwidth solution by running a local SCRIBE-RSS proxy. Existing end-user RSS tools could poll a local SCRIBE-RSS proxy as often as they want without unnecessary bandwidth usage. Users would then see new SCRIBE-RSS items sooner than they would under a more conservative polling policy.

4.4 Discussion

The system we propose offers substantial benefits for both producers and consumers of RSS data. The chief incentive for content providers is the lower cost associated with publishing micronews: large Web sites with many readers may offer large volumes of timely content to SCRIBE-RSS clients without fear of saturating their network links, and a smaller Web site need not fear sudden popularity when publishing a SCRIBE-RSS feed. SCRIBE-RSS also offers publishers an opportunity to provide differentiated RSS services, perhaps by publishing simple (low-bandwidth) headlines in a conventional RSS feed, while delivering full HTML stories in SCRIBE-RSS.

End users will receive even *better* news service with SCRIBE-RSS than is currently possible. While users currently punish Web sites with increasingly aggressive polling schedules in order to get fresh news, no such schedule will match the timeliness of SCRIBE-RSS, in which users will see new items within seconds—not minutes or hours. If publishers begin to offer richer micronews through SCRIBE-RSS, we believe users will be even more likely to use the system. Finally, by building Pastry and Scribe into users' RSS clients, we further improve the performance and scalability of the overlay network, which will in turn improve the performance of micronews dissemination.

With the increasing richness of SCRIBE-RSS content comes the potential incentive for freeloading. A technically sophisticated attacker may develop a client that attempts to receive RSS items from Scribe without redistributing those items, thus preventing other nodes from receiving Scribe events. We do not expect this to be a common attack, since the outbound bandwidth contribution of any single participant in SCRIBE-RSS is expected to be small, but in order to address this problem we can leverage similar concepts from SplitStream [7] which divides the content into multiple trees so that nodes are only required to forward in one of them.

Additionally, incentives-compatible mechanisms to ensure fair sharing of bandwidth [18] can be applied if most users subscribe to several feeds, which is a common model of RSS usage. Since a participant is likely to simultaneously assume parent and child roles in different groups, bandwidth is likely to be balanced between pairs of cooperative nodes. In the case of freeloading, a bandwidth imbalance is detected, and service may be denied to the malicious node.

5 Conclusions and Future Work

The current RSS polling mechanism has been said to scale well because "its cost is almost directly proportional to the number of subscribers" [5]. In fact, linear cost is typically an indicator of poor scaling properties, especially when that cost is focused on one member of a distributed system. It is likely that the further growth of RSS adoption will be badly stunted, or halted entirely, without substantial change to the way micronews is distrubuted.

The proposed SCRIBE-RSS subscription system for RSS takes advantage of the properties of peer-to-peer event notification to address the bandwidth problem suffered by Web content providers, while at the same time bringing micronews to end users even more promptly than is currently possible. Self-organizing subscription systems like Scribe offer scalability that cannot be matched by any system designed around resource polling.

Building upon the SCRIBE-RSS distribution system, we foresee a potential for entirely new services based on RSS which cannot be accomplished today. By using single-writer logs in combination with a distributed

storage mechanism such as a DHT [20, 14, 9], we can record permanently every RSS item published, allowing a distributed archival store of micronews across the Internet. Clients of such a system would easily be able to find out what they "missed" if they have been offline for so long that old RSS items are no longer available in any conventional, static RSS feed. We may also investigate anonymous RSS feeds, by channeling items through an anonymizing peer-to-peer routing system, such as AP3 [15]. Finally, we can envision the use of high-bandwidth multicast (such as SplitStream [7]) to distribute large files—such as software, audio, and video—as part of SCRIBE-RSS feeds.

References

[1] Atom Syndication Format. http://www.atomenabled.org/developers/syndication/.

[2] Bloglines Web Services. http://www.bloglines.com/services/.

[3] NewsGator Online Service. http://www.newsgator.com/ngs/.

[4] RSS protocol (Wikipedia entry). http://en.wikipedia.org/wiki/RSS_(protocol).

[5] RSS for Mac OS X Roundtable. http://www.drunkenblog.com/drunkenblog-archives/000337.html, Oct. 2004.

[6] M. Castro, P. Druschel, Y. C. Hu, and A. Rowstron. Exploiting network proximity in peer-to-peer overlay networks. Technical Report MSR-TR-2002-82, Microsoft Research, May 2002.

[7] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth multicast in cooperative environments. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, Oct. 2003.

[8] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *IEEE JSAC*, 20(8), Oct. 2002.

[9] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-area cooperative storage with CFS. In *Proc. ACM SOSP'01*, Banff, Canada, Oct. 2001.

[10] L. Gomes. How the next big thing in technology morphed into a really big thing. *The Wall Street Journal*, Oct. 2004.

[11] H. Green. All the news you choose – on one page. *BusinessWeek*, Oct. 2004. http://www.businessweek.com/magazine/content/04_43/b3905055_mz011.htm.

[12] M. Hicks. RSS comes with bandwidth price tag. *eWeek*, Sept. 2004. http://www.eweek.com/article2/0,1759,1648625,00.asp.

[13] V. Kopytoff. One-stop way to read news, blogs online: RSS allows users to get free, automatic feeds. *The San Francisco Chronicle*, Oct. 2004. http://www.sfgate.com/cgi-bin/article.cgi?file=/chronicle/archive/2004/10/25/BUG1U9ES301.DTL.

[14] 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 store. In *Proc. ASPLOS'2000*, Cambridge, MA, November 2000.

[15] A. Mislove, G. Oberoi, A. Post, C. Reis, P. Druschel, and D. S. Wallach. AP3: Cooperative, decentralized anonymous communication. In *Proceedings of the SIGOPS European Workshop*, Leuven, Belgium, Sept. 2004.

[16] R. C. Morin. HowTo RSS Feed State. http://www.kbcafe.com/rss/rssfeedstate.html, Sept. 2004.

[17] Netscape Communications Corp. *My Netscape Network*, Mar. 1999. http://www.purplepages.ie/RSS/netscape/rss0.90.html.

[18] T.-W. J. Ngan, A. Nandi, A. Singh, D. S. Wallach, and P. Druschel. On designing incentives-compatible peer-to-peer systems. In *Proc. 2nd Workshop on Future Directions in Distributed Computing*, Bertinoro, Italy, June 2004.

[19] M. Pilgrim. The myth of RSS compatibility. http://diveintomark.org/archives/2004/02/04/incompatible-rss, Feb. 2004.

[20] A. Rowstron and P. Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proc. ACM SOSP'01*, Banff, Canada, Oct. 2001.

[21] R. Scoble. A theory on why RSS traffic is growing out of control. http://radio.weblogs.com/0001011/2004/09/08.html#a8200, Sept. 2004.

[22] W. Vogels. Once more: Polling does not scale. http://weblogs.cs.cornell.edu/AllThingsDistributed/archives/000511.html, Sept. 2004.

[23] N. Wallace. RSS is sticky traffic. http://www.synop.com/Weblogs/Nathan/PermaLink.aspx?guid=db37ec96-9271-4e4a-ad8d-6547f27fc1cb, July 2004.