

Automatic Subscriptions In Publish-Subscribe Systems*

Lars Brenna Cathal Gurrin Dag Johansen Dmitrii Zagorodnov

Dept of Computer Science
University of Tromsø, Norway
{larsb, gurrin, dag, dmitrii}@cs.uit.no

Abstract

In this paper, we describe how to automate the process of subscribing to complex publish-subscribe systems. We present a proof-of-concept prototype, in which we analyze Web browsing history to generate zero-click subscriptions to Web feeds and video news stories. Our experience so far indicates that user attention data is a promising source of data for automating the subscription process.

1 Introduction

Publish-subscribe systems offer an efficient and convenient medium for one-to-many distribution of time-sensitive data. In our work on augmenting the traditional pull-based Internet with push-based information filters [11], time-sensitive data consists of notifications about new or changed information on the Web. Our experience shows that with such heterogeneous data, specifying subscription queries that return only relevant items is challenging. In the worst case, having to manage subscriptions manually – devising appropriate keywords, refining the query to control volume of updates, unsubscribing to queries that are no longer relevant – can discourage users from using a notification system.

We conjecture that utility of publish-subscribe systems in many domains, especially those with heterogeneous data, can be increased by automating the management of subscriptions. The automation can come from monitoring the behavior of users – the Web sites they visit, the documents they read, the texts they write – and feeding the logged data to recommendation services. Well-known information retrieval techniques, including correlation of logs from multiple users, can

be used to produce both topic- and content-based subscriptions. By delegating to a recommendation service the task of creating, refining, and removing subscriptions in a publish-subscribe system, the user can receive relevant information without any additional effort.

The goal of this paper is to show how subscriptions in publish-subscribe systems can be automated with a novel application of information retrieval techniques. In the next section we outline the components of a monitoring and recommendation architecture, called Reef, that we have used to generate automatic subscriptions. Reef is geared toward monitoring Web browsing history and making recommendations to a user through a browser extension. We then present, in Section 3 and Section 4, a centralized and a distributed version of such an architecture, along with two case studies. Specifically, we discuss recommendation of Web feeds stories and video news stories. Finally, we conclude the paper after discussions of related work in Section 5.

2 Better Surfing on the Reef

Reef is an architecture that we propose for automating the subscription process in publish-subscribe systems. Since functionality of a system depends on its architecture, we have considered both centralized and distributed designs. We begin by defining what functionality is needed and go on to describe our experimental designs and practical experience.

2.1 From Attention to Subscriptions

The gap between people’s interests expressed in a natural language and subscriptions expressed in an event algebra, such as Cayuga [6], is large. Event algebra is useful for expressing complex event triggers succinctly, but we cannot expect all Internet users to express their interests this way. Hence, we propose a system that manages subscriptions automatically. In the extreme case, the only input to this system can

*This work is sponsored by the Norwegian Research Council Programmes No. 162349 and IKT-2010.

be *user attention*, which is an encoding of some of the actions that the user performs.

Consider a publish-subscribe system with a well-defined event algebra syntax and a specification for valid name-value pairs in the system. In our approach, we analyze the continuous stream of user attention, looking for tokens that can form valid name-value pairs for the publish-subscribe system in question. We conjecture that a system can be built that is general enough for use with any well-defined publish-subscribe interface.

2.2 Basic Functionality

The architecture of Reef can be broken down into four components.

The attention of a user is captured by an *attention recorder*. In our prototype, the recorder runs in the Web browser and captures the URIs viewed by the user. Potentially, the recorder could capture the characters typed by the user, the text in the files the user viewed, the meta-data of sound files played, etc.

This raw data is processed by an *attention parser*, which looks for tokens that match the specification of name-value pairs of the publish-subscribe system we are given. For example, in a publish-subscribe system that delivers stock quotes, the attention parser would be looking for known stock symbols in the attention data. Other examples of tokens are: feed URLs, which can be used in Web feed subscriptions; or any commonly occurring keywords, which can be used in many content-based systems, such as the news video recommendation system that we describe in Section 3.3.

Using the tokens found by the parser, a *recommendation service* makes recommendations on what subscriptions to place and which to remove. In response, a *subscription frontend* activates or deactivates subscriptions, as well as receives and displays the events that arrive.

Whether the user appreciates the recommendations or not is determined by his attention to the delivered events. For instance, clicking of a link contained in an event will be captured by the attention recorder and can be viewed by the recommendation service as positive feedback. The result is a “closed-loop” system that requires *no* explicit user feedback (although it does not preclude it, either).

3 Centralized Reef

Figure 1 shows a design in which a centralized server builds up a database of attention data (transferred in step 1) for each user. The server analyzes the attention

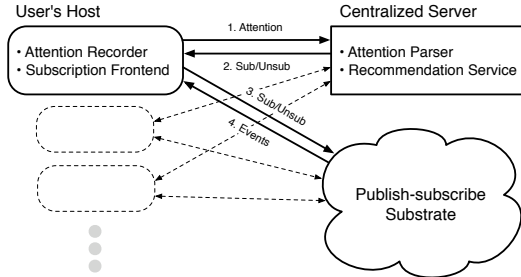


Figure 1. Architecture for a centralized subscription recommendation system.

data to recommend subscribe/unsubscribe actions to the subscription frontend (2). The latter executes the actions (3) and receives events (4) directly from the publish-subscribe substrate.

The centralized solution resembles a large-scale search engine in that it indexes a lot of data on behalf of many users. Such large data collections are fit for many data mining applications such as collaborative subscription recommendations across applications, mediums, and users. One downside is the high cost of scaling to many users. Another is the threat to privacy. The recommender will implicitly gain extensive knowledge on the habits and interests of every user.

In our implementation, a back-end server takes care of storage and computations, whereas users only need to install a browser extension (a Firefox plug-in written in Javascript) to record attention data and present incoming events. The server system is a standard “LAMP” setup: Linux, Apache, MySQL, PHP, and Python. Components that receive attention data from users and manage cookies are written in PHP, while the more complex application logic is written in Python.

For compatibility with the current Web, subscriptions are deployed at WAIF Proxies as described in [2]. This approach makes it possible to “wrap” and extend any pull-based resource, such as an RSS feed, with a push-based interface. Hence, we make our recommendation system backwards compatible with current Internet services.

3.1 Data Collection and Analysis

Our attention recorder, implemented as a browser extension, logs every outgoing HTTP request and periodically forwards batches of requests to a Reef server. Several attributes, such as a timestamp and a user cookie, are logged along with the URI of the request. This unit of attention data is called a *click*. The cookie allows the server to tie a click to a user.

One or more attention parsers and subscription recommenders run on the Reef server. When clicks arrive, they are stored in a database and the URIs in them are batched for periodic crawling. The crawler retrieves the pages that the users visited and analyzes them in several ways: It looks for ad servers and spam sites, as well as multimedia, and flags them as such in the database, ensuring they will not be crawled again. It scans the pages looking for sources of Web feeds. It also parses the page to extract common keywords, as will be discussed in Section 3.3. The Reef server then makes subscription recommendations to the browser extension.

When the browser extension receives a server’s recommendation, it automatically places that subscription. The subscription will remain in place either until the user unsubscribes from it or until the server recommends that the extension does so. The events from subscriptions are displayed in a sidebar (a vertical panel to the left of the browsing panel) either as text or as a small image with annotation. The user may click on the event to view it in the browsing panel or click on a button to delete it. If the user ignores the event for a certain period of time, it expires and disappears from the list.

3.2 Topic-based Subscriptions

Attention data can be parsed to find topics for topic-based subscriptions. For a case study, we experimented with automatic subscriptions to RSS feeds based on the browsing history of users.

When a new RSS feed is discovered by the crawler among the pages visited by the user, the address of the feed is sent as a recommendation to the user’s browser extension. The extension then subscribes to the feed using the WAIF FeedEvents service [2]. This service is a push-based proxy for RSS feeds. It can poll any RSS, Atom, or RDF feed, and check for updated content on behalf of many users.

Using ten weeks of browsing history from five test users, we recorded over 77000 requests to 2528 distinct Web servers. 70% of the requests were to 1713 advertisement servers, and 807 servers were visited only once. On the remaining 906 Web servers, 424 distinct RSS feeds were found.

On one hand, this demonstrates that our technique can reveal many potential sources of topic-based subscriptions. On the other hand, some filtering of results is clearly necessary. Even though most feeds are updated infrequently [13], we still found enough feeds to overwhelm any user with updates. We are currently investigating approaches to using attention data for fil-

tering of updates and for removing subscriptions.

3.3 Content-based Subscriptions

Attention data can be analyzed to form many types of content-based subscriptions. The general idea is to extract the most common keywords from the attention data and to build simple queries out of them. For a case study, we experimented with queries for video news stories based on the browsing history of users.

From a log of six weeks of Web browsing by a test user, we extracted the most important terms from over 10,000 pages visited by the user in that period¹ and used the top N of them to form content-based queries. (We varied N between 5 and 500.) The queries determined the order in which news stories were returned from an archive of 500 video stories² that aired on ABC and CNN in 2004. After collecting the browsing history, we asked the test user to rank the stories by interest, which allowed us to measure how effective the query was at placing the most interesting stories first as compared to the order in which the stories originally aired.

Our initial results show that the query increases the precision of recommended content regardless of the number of terms used to construct it. We found that the optimal number of terms required was 30, with which the precision peaked at 34% improvement, meaning that a third more interesting stories appeared in the front. With only five terms, precision improved by 12%. These results are promising and we are currently engaged in a wider study, with greater number of users and more documents in the video archive.

We have not yet addressed the problem of forming queries for users that have many diverse interests. Rather, we are relying on the term weighting mechanism for selecting the 30 terms that sufficiently encompass a user’s general interests. Further experimentation will be required to identify what improvement in precision can be gained by considering diversity of interests in a user’s history.

4 Distributed Reef

Limitations of the centralized subscription recommendation system prompted us to consider a dis-

¹We chose terms using a modified version of Robertson’s Offer Weight formula [16] which integrates the term frequency measure into the ranking process.

²The stories came from a video dataset used at the TRECVideo workshop (<http://www-nlpir.nist.gov/projects/trecvid/>). For ranking, we used the BM25 algorithm [16] with parameters trained from a previous experiment [9] into user relevance feedback for video search.

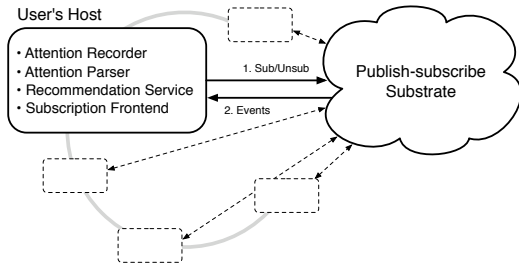


Figure 2. Architecture for a distributed peer-to-peer subscription recommendation system.

tributed peer-to-peer design, shown in Figure 2. In this configuration, the attention data stays on the user’s host, where the subscription recommendation software analyzes it. As in the centralized design, the subscription frontend performs subscribe or unsubscribe operations (step 1) on a publish-subscribe substrate. Events are delivered to the subscription frontend (2).

Correlating the interests of users to generate collaborative recommendations is a more complex task in the distributed model. However, peers can be grouped for the exchange of recommendations using collaborative techniques that will be described in Section 5.2.

There are several advantages to the distributed design. Storage and computational load will be distributed among the participants, allowing the system to scale naturally and eliminating the single point of failure. Furthermore, when the attention parser and the recommendation service reside on the user’s host, crawling of documents fetched by the user is typically unnecessary as they may be available from the browser’s cache. Thus, network load is reduced.

Running the recommendation service on the user’s host also gives the user full control over the attention data. The recommendations can be based on data that the user is reluctant to share with a centralized service, which can improve precision.

5 Related Work

Work related to our project comes from several different research areas, as well as from deployed online services.

5.1 Recording Attention

Recording feedback from users is an increasingly common practice among online services, for some of which the feedback is critical.

Much feedback is submitted voluntarily, through comment forms that rate a product or an organization (e.g. *Amazon* book reviews and *eBay* merchant ratings) or through application-specific interfaces (e.g. thumbs up/down buttons of *Stumble Upon*). While such explicit feedback can be precise [19], not all users bother to submit it and it is not equally useful in all domains.

Morita & Shinoda [14] achieved better content filtering through implicit feedback by measuring the time spent reading individual news stories. Their findings were largely confirmed in the evaluation of the *Curious Browser* [5]. There are many examples of services collecting such implicit feedback, sometimes without the user being aware of it. *AudioScrobbler* and *iTunes* record the songs played by users to enable recommendations and popularity ranking. To improve search precision, *Google* uses many types of attention data from users that rely on their services for Web and local disk searching, Web publishing, email, etc.

Collection of personal information, no matter for what cause, threatens the privacy of the users involved. The founders of *AttentionTrust*³ advocate that the attention data belongs to the user. The operators of *Root*⁴ acknowledge that the user should have full control over who profits from the use of attention data. In the distributed Reef, the user is in control of the attention data.

5.2 Analyzing Attention

Once collected, attention data can be utilized to improve the quality of search results or recommendations by employing a variety of information retrieval techniques. Preliminary research examined the analysis of a user’s interaction with hypermedia systems in order to personalize content and link presentation. It has been shown experimentally [12] that utilizing user context can increase the speed of user navigation of a hypertext system.

On the Web, the analysis of a user’s browsing history in order to generate long-term or short-term models of a user’s interests relies on the construction of user interest models, which are employed to personalize search results for each user, or to improve recommendation of products or services in e-commerce. While conventional search engines are unlikely to maintain a user model for each user, to improve the quality of document ranking, attention data is used to provide other related services (as seen in Section 5.1). However, the application of attention data to improve search engine

³<http://www.attentiontrust.org>

⁴<http://root.net>

rankings, by utilizing group-based profiling, has been shown to be beneficial for personalizing search engine rankings in the I-SPY search engine [18]. I-SPY operates by analyzing user search behaviour, which is repetitive and regular in nature, where similar queries and results are selected by similar users. Instead of maintaining a user profile, a group profile is being maintained which groups users into communities with similar interests and utilizes the past searching behaviour (attention data) of the group as a whole to rerank the output of a conventional search engine.

The novelty in our approach is an architecture that allows using these techniques in a new setting, namely automatic subscriptions in publish-subscribe systems.

5.3 Pushing Notifications

Web feed formats, such as RSS and Atom, have become popular means for timely notification of updates to Web sites. However, current implementations rely on direct connections between clients and the server, so frequent pulling from many users strains network and server resources with unnecessary traffic [13]. Feedtree [17] and Corona [15] create peer-to-peer feed distribution networks that scale through collaborative pulling.

In contrast to Web feeds, full-fledged publish-subscribe systems, such as Siena [3], SCRIBE [4], and Gryphon [1], mainly focus on efficient event dissemination. These systems deal with the trade-off between scalability and expressiveness in a distributed environment. Some centralized systems, such as YFilter [7] and Cayuga [6], manage to have very expressive event algebra while maintaining extreme scalability; the latter allows stateful subscriptions which span multiple events, as well as parametrization and aggregation. These two systems, however, do not address event routing.

None of the systems mentioned aid users in constructing subscriptions.

5.4 Presenting Notifications

The challenge in presenting notifications lies in making sure they are noticeable but not disturbing.

Web-based notification services typically embed suggestions into the pages that they return (e.g. “Customers who bought this also bought...”), which ensures that notifications do not appear at inappropriate times. Projects like “Stuff I’ve Seen” [8] and Haystack [10] investigate efficient approaches to finding and displaying information that a user has seen in the past. *Google*

*Desktop*⁵ uses a sidebar to display information related to the user’s current tasks. In particular, it will automatically display Web feeds from recently visited Web sites, just as we do in our experiment.

6 Conclusions

Our motivation for this work was that users tend to visit the same information sources repeatedly, looking for updates. In our work, we essentially imitated that behaviour by placing subscriptions to updates of some of the sources frequented by the user. We thus hope to discover relevant updates and deliver them in a timely manner.

It can be argued that expressive and scalable event dissemination systems have made it possible for publishers of short-lived and urgent information to forward events efficiently to many users. However, such systems typically use complex query languages that are meaningful only to experienced programmers. In fact, even subscribing to Web feeds is not straightforward because of different formats and because different applications treat feed URLs differently.

Reef demonstrates automatic recommendations of simple topic-based subscriptions for Web feeds and content-based subscriptions for video news stories. To do this, Reef records and analyzes the attention data of users. Our results indicate that this is a promising method for subscription recommendations. On average, every user received one new feed recommendation per day during our test period. Precision of video news delivery could be improved by 34%.

New attention recorders and parsers can be integrated into our architecture. Hence, many types of data available on a user’s computer can be used to recommend subscriptions to relevant information in publish-subscribe systems. We have demonstrated that, given a publish-subscribe interface, it is possible to generate suitable subscriptions from user attention data. Although the idea of using attention data for recommendations is commonly used in contemporary online services, using it to improve the usability of publish-subscribe systems, is novel.

Acknowledgments

The authors thank Keith Marzullo, Ingar M. Arntzen, Håvard D. Johansen and Ole-Petter Wikene for valuable discussions and support. We would also like to thank the anonymous referees for many helpful comments.

⁵<http://google.com/desktop/>

References

- [1] G. Banavar, T. D. Chandra, B. Mukherjee, J. Nagarajarao, R. E. Strom, and D. C. Sturman. An efficient multicast protocol for content-based publish-subscribe systems. In *ICDCS '99: Proceedings of the 19th IEEE International Conference on Distributed Computing Systems*, page 262, Washington, DC, USA, 1999. IEEE Computer Society.
- [2] L. Brenna and D. Johansen. Engineering push-based web services. *International Journal of Web Services Practices (IJWSP)*, 1(1):89–100, 2006.
- [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 (TOCS)*, 19(3):332–383, Aug. 2001.
- [4] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication (JSAC)*, 20(8), Oct. 2002.
- [5] M. Claypool, P. Le, M. Wased, and D. Brown. Implicit interest indicators. In *IUI '01: Proceedings of the 6th international conference on Intelligent user interfaces*, pages 33–40, New York, NY, USA, 2001. ACM Press.
- [6] A. Demers, J. Gehrke, M. Hong, M. Riedewald, and W. White. Towards expressive publish/subscribe systems. In *Proceedings of the 10th International Conference on Extending Database Technology (EDBT 2006)*, page (To Appear), March 2006.
- [7] Y. Diao, M. Altinel, M. J. Franklin, H. Zhang, and P. Fischer. Path sharing and predicate evaluation for high-performance xml filtering. *ACM Trans. Database Syst.*, 28(4):467–516, 2003.
- [8] S. Dumais, E. Cutrell, J. Cadiz, G. Jancke, R. Sarin, and D. C. Robbins. Stuff i've seen: a system for personal information retrieval and re-use. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 72–79. ACM Press, 2003.
- [9] C. Gurrin, D. Johansen, and A. F. Smeaton. Supporting relevance feedback in video search. In *Proceedings of the 28th European Conference on Information Retrieval (ECIR 2006)*, page (To appear), April 2006.
- [10] D. Huynh, D. Karger, and D. Quan. Haystack: A platform for creating, organizing and visualizing information using rdf, 2002.
- [11] D. Johansen, R. van Renesse, and F. B. Schneider. WAIF: Web of asynchronous information filters. In A. Schiper, A. A. Shvartsman, H. Weatherspoon, and B. Y. Zhao, editors, *Future Directions in Distributed Computing*, volume 2584 of *Lecture Notes in Computer Science*, pages 81–86. Springer, 2003.
- [12] C. A. Kaplan, J. Fenwick, and J. Chen. Adaptive hypertext navigation based on user goals and context. *User Model. User-Adapt. Interact.*, 3(3):193–220, 1993.
- [13] H. Liu, V. Ramasubramanian, and E. G. Sirer. Client and feed characteristics of rss, a publish-subscribe system for web micronews. In *USENIX Internet Measurement Conference (IMC)*, New Orleans, Louisiana, Oct. 2005.
- [14] M. Morita and Y. Shinoda. Information filtering based on user behavior analysis and best match text retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 272–281, New York, NY, USA, 1994. Springer-Verlag New York, Inc.
- [15] V. Ramasubramanian, R. Peterson, and E. G. Sirer. Corona: A high-performance publish-subscribe system for web micronews. In *in Networked System Design and Implementation (NSDI'06)*, page (To appear), May 2006.
- [16] S. Robertson and K. Jones. Simple proven approaches to text retrieval, 1997.
- [17] D. Sandler, A. Mislove, A. Post, and P. Druschel. Feedtree: Sharing web micronews with peer-to-peer event notification. In *Proceedings of the 4th International Workshop on Peer-to-Peer Systems (IPTPS'05)*, Ithaca, New York, Feb. 2005.
- [18] B. Smyth, E. Balfe, J. Freyne, P. Briggs, M. Coyle, and O. Boydell. Exploiting query repetition and regularity in an adaptive community-based web search engine. *User Model. User-Adapt. Interact.*, 14(5):383–423, 2004.
- [19] T. Yan and H. Garcia-Molina. SIFT—A tool for wide-area information dissemination. In *Proc. 1995 USENIX Technical Conference*, pages 177–186, New Orleans, 1995.