WebFilter: A High-throughput XML-based Publish and Subscribe System

João Pereira*
INRIA Rocquencourt
Joao.Pereira@inria.fr

Françoise Fabret
INRIA Rocquencourt
Francoise.Fabret@inria.fr

H. Arno Jacobsen
Toronto University
jacobsen@eecg.toronto.edu

François Llirbat
INRIA Rocquencourt
Francois.Llirbat@inria.fr

Dennis Shasha†
Courant Institute of Mathematical Sciences
New York University
shasha@cs.nyu.edu

1 Overview

Large-scale information dissemination systems for selective information distribution are gaining increasing importance. This is due to the fast proliferation of different communication infrastructures, e.g., Internet and wireless networks, and the increase in mobile computing potential. However, targeted information dissemination will only be effective if the distributed information is pertinent for the user. In this context the publish and subscribe paradigm is a good interaction model: This interaction model enables publishers to publish information through the publish and subscribe system and provides subscribers with the ability to express their interest within the system in order to be notified afterwards of any publications matching their registered interests.

In the context of information dissemination, requirements for a publish and subscribe (pub/sub) system are threefold: First, the system must achieve interoperability between different devices and infrastructures; One solution is to use eXtensible Markup Language (XML) as the common data presentation format for publications. Second the system must support a (very) expressive filtering language that enables subscribers to accurately define their interests over publications. Finally, it has to be able to manage millions of users, manage high volatility of users' interests, and filter large quantities of content per unit of time.

We have built a pub/sub system, called WebFilter, that aims at large-scale and high-throughput XML processing for selective information dissemination on the Internet and in wireless environments. The main characteristics of our system are the following:

1. A *publication* to our system may be any well-formed *XML-document*.
2. A *subscription* to our system is an *XPath-expression* [9] as defined by the W3C. XPath treats an XML document as a tree of nodes. XPath provides a flexible way to specify tree patterns that have to be matched by XML documents.
3. *Scalable* and extremely *efficient matching*, WebFilter can manage about *1 Million* XPath-expressions and *70* XML-documents per second.
4. WebFilter is *dynamically updatable*, i.e., subscriptions may be efficiently inserted, updated, and deleted while the system is in operation.

A large number of information dissemination systems have been developed in the context of Message Oriented Middleware and Selective Dissemination of Information communities. In most of these systems, events consist of sets of (attribute, value) pairs and are filtered according to their attribute values [7, 6, 2, 3, 8]. Only a small number of systems [5, 1] supports XML filtering as WebFilter does. Compared to these systems, WebFilter achieves much better performance.

2 System description

The architecture of our system is shown in Figure 1. It processes XML-documents by translating them into

*The work of J. Pereira was partly supported by a JNICT fellowship of Program PRAXIS XXI (Portugal).

†The work of D. Shasha was partly supported by grant 9988345 of the United States National Science Foundation.

Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

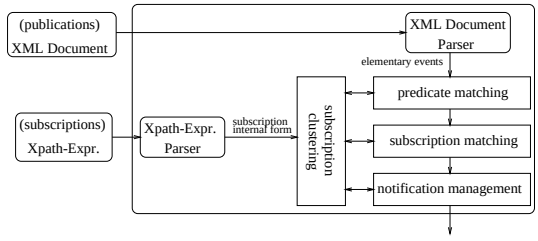

Figure 1: System architecture.

a set of *elementary* events (one *elementary* event per path in the document). An elementary event describes the structure of a path in terms of elements, relative positions of elements, element attributes and associated values. Subscriptions are also compiled in an internal form and stored in the memory of the matching engine. Moreover, we use clustering strategies [3] that group subscriptions being likely to match together on similar elementary events. This limits the number of subscriptions to consider at each event. Then, we match the structure of the document as a tree against the subscriptions that are matched by at least one elementary event. The core filter engine and the XML and XPath processing are implemented in C. Notification processing and user interface to the system is implemented in a combination of Java with HTML browser-based visualization.

3 Demonstration: Selective dissemination of ads

3.1 Application

To show the main features of WebFilter, we have developed a selective dissemination of advertisements application using a DTD [4] defined by the NAA Classified Advertising Standards Task Force [4]. Figure 2 depicts a fragment of an XML document that is valid with respect to this DTD. This fragment describes several pieces of information, some relevant to the media distributor, some relevant to the media consumer.

```
<ad-instance>
  ...
  <real-estate>
    <house>
      <size length=''20''  width=''15'' .../>
      <plan plan-id=''12232''  plan-src=''CAD''/>
      <bedrooms number=2 >
         Two bedrooms with view of lake
      </bedrooms>
      ...
    </house>
    <support-info>
      <file-id>  253432 </file-id>
      ...
    </support-info>
  </real-estate>
  ...
</ad-instance>
```

Figure 2: Fragment of the NAA DTD

In the demonstration, a user can submit subscriptions that identify her interests in certain subjects by writing XPath expressions using the web interface of our system. For example, users can submit the following query: $//real\text{-}estate/*/bedrooms[@number = 2]$.

In this subscription, the subscriber expresses interest in real-estate objects of unspecified kind (i.e., expressed with the wildcard operator "*") with specifically constrained number of bedrooms. This XPATH expression matches XML documents which contain a `real-estate` element followed by any other level in the document, followed by a `bedrooms` element with `number` attribute equal to 2.

3.2 Performance

We demonstrate the performance of WebFilter using the benchmark proposed in [1]. Our experiments show that our system can achieve high throughput even for large numbers of subscriptions. For instance, we have run an experiment where submitted XML documents have an average depth of 4.4 and an average number of 26.3 elements and subscriptions are path expressions having an average depth of 3.34 and contain one filter expression that defines an equality condition over an element attribute. This experiment has been executed on a single-CPU workstation with an i686 CPU at 500MHz and 256MB RAM operating under Linux. The experiment shows that our system can process 70 documents per second for one million of subscriptions.

Acknowledgements: We would like to thank Mehmet Altimel and Michael J. Franklin for making their XML filtering benchmark tool available to us.

References

[1] M. Altinel, D. Aksoy, T. Baby, M. Franklin, W. Shapiro, and S. Zdonik. Dbis toolkit: Adaptable middleware for large scale data delivery (demo description). In *Proceedings of the ACM SIGMOD*, 1999.

[2] A. Carzaniga, D. Rosenblum, and A. Wolf. Achieving scalability and expressiveness in an internet -scale event notification service. In *Nineteenth ACM PODC*, 2000.

[3] F. Fabret, A. Jacobsen, F. Llirbat, J. Pereira, K. Ross, and D. Shasha. Filtering algorithms and implementation for very fast publish/subscribe systems. *Proceedings of the ACM SIGMOD conference*, 2001.

[4] http://www.naa.org/technology/clsstdtf. *NAA CLASSIFIED ADVERTISING STANDARDS TASK FORCE.*

[5] http://www.xmlblaster.org/.

[6] IONA Technologies. *http://www.iona.com.*

[7] New Era of Networks. *http://www.neonsoft.com.*

[8] B. Nguyen, S. Abiteboul, G. Cobena, and M. Preda. Monitoring xml data on the web. *Proceedings of the ACM SIGMOD conference*, 2001.

[9] W3C http://www.w3.org/TR/xpath.html. *XML Path Language (XPath), W3C recommendations.*