XML Routing in Data Dissemination Networks

Guoli Li   Shuang Hou   Hans-Arno Jacobsen
University of Toronto, Canada {gli@cs, shou@cs, jacobsen@eecg}.utoronto.ca

1. Introduction

XML-based data dissemination networks are starting to become a reality. In a dissemination network, data, marked-up in XML, is routed based on filter expressions stored at intermediate nodes that indicate where the XML document is to be routed to. Filter expressions, often expressed as XPath expressions (XPEs), are submitted by data consumers who express interest in receiving certain kinds of documents. For instance, a globally operating insurance company with many branch offices distributed world-wide is linked by an overlay network of content-based routers that comprise the XML dissemination network. An insurance claim, an insurance bid, or a request for proposal can be submitted anywhere into the overlay network (e.g., by a third party insurance broker or an online client) and be routed toward a currently online, specific expert employee, speaking the same language as the requester. Note, the latter constraints are expressed as XPE filter expressions against which the XML document is evaluated in transit. This design fully decouples information requesters and information providers, avoids a single point of control and a single point of failure, and increases scalability due to decentralization and distribution.

This paper addresses the XML/XPath content-based routing problem. More specifically, this paper focuses on the problem of efficiently routing an XML document emitted from a data producers at one point in the network to a set of data consumers located anywhere throughout the network. Prior to receiving XML documents, consumers must have expressed interest in receiving XML documents by registering XPEs with the network. This problem statement is akin to the well-known publish/subscribe matching problem. However, the main difference here is that in the case of data dissemination networks there exists no one single centralized publish/subscribe system, but a network of content-based routers (i.e., a network or federation of publish/subscribe systems.) In the dissemination network, XML documents are routed based on their content and not based on IP address information, which is, due to the completely decoupled design, not available – all routing decisions are exclusively based on content information. Fig. 1 provides an overview of the dissemination network this paper assumes. In the overlay network depicted in Fig. 1 each broker only knows their neighbors (i.e., in terms of IP network address information.) However, none of the clients – neither data producers nor data consumers know about each other or about the network topology, except the router they connect to.

In the context of XML-based data dissemination, one of the main challenges is the ability to efficiently deliver relevant XML documents to a large and dynamically changing group of consumers. Centralized XML filtering [5, 3, 9] and distributed query-based XML retrieval approaches [6, 11, 10] have found wide-spread interest, but do not address the distributed, content-based routing problem articulated above and addressed in this paper. Content-based routing [2, 16, 4, 14, 7] in distributed publish/subscribe architectures, have been studied for non-XML-based data. Their operational model assumes sets of attribute-value pairs joined by Boolean operators. It is not at all obvious how to extend these approaches for semi-structured data, especially due to the hierarchical data model of XML.

Our own prior research on content-based routing [14, 7, 12] develops architectures, algorithms and protocols for content-based routing of non-XML based data. It is a non-trivial problem, as we argue in this paper, to extend these approaches to the hierarchical structure of XML. Our prior work on developing efficient matching algorithms for XML [9] addresses part of the problem, but does not address the important problem of efficiently computing routing decisions, inducing advertisements from XML DTDs, and determining covering and merging relations, which is the focus of this research. This paper combined with our earlier work on XML matching comprises all components required to build an efficient content-based router for an XML data dissemination network.

In this paper, we sketch algorithms for dissemination of XML data throughout a network of content-based routers towards data consumers who have specified their interests through XPEs. The full details of our algorithms are presented in [13]. We adapt the use of advertisements to optimize data dissemination from non-XML-based routing systems. While this idea is common in the publish/subscribe literature [2, 16, 4, 14, 7], it is not known how to extend these concepts to the data model underlying XML. We demonstrate how to use the XML Document Type Definition (DTD) to generate advertisements about the information a data producer is going to publish. We develop advertisement-based routing algorithms and propose a novel data structure to maintain XPEs by identifying the covering relations between them. We present

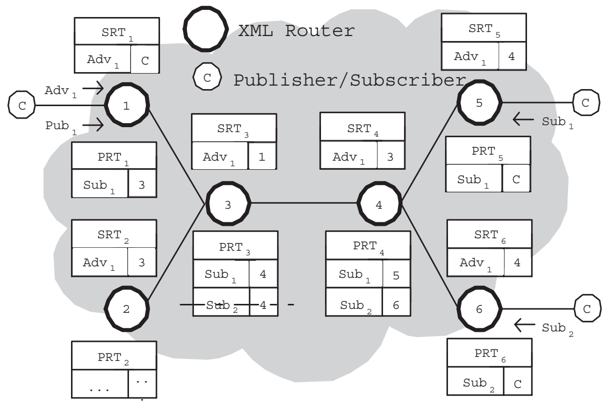

Figure 1. Content-based Routing

covering algorithms for XPEs to reduce the routing table size stored at each router and speed up routing computation. We present an optimization of merging similar XPEs to further reduce routing computations.

2. Content-based Routing

Content-based publish/subscribe systems [2, 16, 4] provide a flexible and extensible environment for information exchange. Messages in content-based pub/sub systems are routed based on their contents rather than the IP address of their destinations. In order to handle a large amount of dynamic information and reduce the network traffic, many optimization techniques, such as advertisements [2] and covering and merging techniques [4, 2, 16], have proven to gain significant benefits for non-XML based publish/subscribe systems. While conceptually, these ideas apply to XML-based data as well, it is not obvious how to apply these concepts to XML. This is the challenge addressed by this paper.

In advertisement-based publish/subscribe systems, advertisements are specifications of information that the publisher will publish in the future. Advertisements are flooded in the publish/subscribe overlay. The common assumption is that the number of advertisement is much less than that of subscriptions and publications. Advertisements are used to avoid broadcasting subscriptions in the network, so that subscriptions are only routed to the publishers who advertise what the subscribers are interested in. Subscriptions define filters on publications. Later on, only matching publications will be delivered to subscribers along the paths built by subscriptions. Fig. 1 shows a scenario for advertisement-based content-based routing. The subscription routing table (SRT) consisting of <advertisement, last-hop > tuples stores advertisements in order to route subscriptions. Publications will trace back along the path setup by subscriptions to interested subscribers. The publication routing table (PRT) maintains the path information. For example, in Fig. 1, adv_1 is broadcast in the network, and is stored on each broker of the network with a different last hop. Consequently, subscriptions that match adv_1 will be routed according to these last hops (e.g., sub_1 is routed along the link $5-4-3-1$). Note that the sub_1 will not be forwarded to brokers 2 and 6 since the adv_1 indicates that the publisher is from broker 1. Therefore, the pub_1 is routed along a reverse path $1-3-4-5$ to the subscriber. In the rest of this paper, we use the notations $P(s)$ and $P(a)$ to refer to the set of publications that match subscription s and advertisement a, respectively.

The goal of covering-based routing [2, 16] is to remove redundant subscriptions from the network in order to maintain a compacted routing table and reduce the network traffic. In Fig. 1, if sub_1 covers sub_2, at broker 4, sub_2 will not be forwarded to broker 3. That is, we can safely remove sub_2 on broker 3 and gain a compacter routing table while maintaining the same information delivery behavior, for all the publications matching sub_2 must match sub_1. Since advertisements have the same format as subscriptions, the covering relations among advertisements can be defined in the same way.

If two subscriptions do not have covering relations, but their publication sets overlap each other, the two subscriptions can be merged to a more general subscription, which covers the original subscriptions. Suppose sub_m is a merger of sub_1 and sub_2, then we have P(sub_m) $\supseteq$ P(sub_1) $\cup$ P(sub_2). There are two kinds of mergers. If the publication set of the merger is exactly equal to the union of the publication set of the original subscriptions, the merger is a *perfect merger*; otherwise, if P(sub_m) $\supset$ P(sub_1) $\cup$ P(sub_2), it is an *imperfect merger*. After merging, only the merger is forwarded into the network. The merging technique [16] is used for further minimizing the routing table size. It is an extension of the covering technique.

3. Related Work

A large body of work has focused on developing publish/subscribe-style matching algorithms for evaluating an XML message against a set of XPEs [5, 3, 1]. However, all these approaches exclusively address centralized matching architectures, not the distributed, content-based XML dissemination networks we address in this work. While matching is an integral step in a content-based router, other routing operations studies in this paper are equally important in a distributed data dissemination architecture. Thus, our work complements matching algorithms for the design of a content-based XML router.

Advertisement-based techniques for optimizing content-based routing have been developed in the area of distributed publish/subscribe [2, 16, 4]. It has been demonstrated that the network traffic and routing table size can be reduced by using different routing strategies, including advertising, covering and merging techniques. However, the main differences between these approaches and our approach lie in the subscription language and publication data format. Our approach is based on the hierarchical, tree-based XPath and XML model; while the traditional content-based routing approaches operate with attribute-value pairs and predicate constraints over these pairs. The advertising carried out by XML and XPath data sources is different and more complex than predicate-based languages, for the hierarchical and recursive structure of the model needs to be taken into account. A DTD of an XML document does not have an equivalent in traditional publish/subscribe approaches. Galanis *et al.* [8] explore XML data dissemination based on a DHT. They use data summaries to ensure that queries are only sent to relevant servers. These data summaries could be taken as a form of advertisements. The data summary is generated from a XML document, so the expressiveness of the data summary is limited to part the DTD. Our contribution is to generate a complete advertisement set once from a DTD for all related XML documents.

Recent research has focused on XML data dissemination [10, 6, 11]. Koloniari *el al.* [10] present a decentralized approach for XML dissemination in a peer-to-peer network. However, in their approach queries are severely restricted in that no wildcards are allowed. Koudas *et al.* [11] propose a flexible routing protocol for XML routers to enable scalable XPath query and update processing in a data-sharing peer-to-peer network. Both approaches are solutions to the *location problem*. The location problem states that given a dynamic collection of XML database servers and an XPath query, find the databases that contain data relevant to the query. Our approach evaluates an XML document against a set of XPath queries, and decides where to route the XML document. Diao *et al.* [6] aim at deploying XML-based services on an Internet-scale, and provide a detailed architectural design for such a system. The NFA used in this work for matching naturally supports a form of specialized covering as XPEs with a common prefix share the NFA path taken while evaluating an input XML message against the XPEs indexed. In addition to prefix covering our approach identifies all cover relations among the XPEs processed and completely eliminates all covered XPEs from the routing computation. Moreover, our work introduces merging and advertising for XML data not addressed in the earlier approach.

4. Content-based XML Routing

In terms of XML document and XPath query routing, the concepts of advertisement, covering, and merging have not been studied. In this section, first, we present the definition and format of advertisement for XML content and give some examples. We also describe how the covering and merging techniques are applied in this context. The detailed discussion can be found in [13].

Definition of Advertisement In the context of XML/XPath routing, the advertisement is generated by exploiting DTD information. The purpose of a DTD is to define the legal building blocks of an XML document. The main building blocks of an XML documents are elements surrounded with tags, e.g., $<$ *root* $>$ ... $<$ */root* $>$. *root* is the element name in the documents. All elements appearing in the XML document must be defined in the corresponding DTD, which determines the structure of elements and their sequence in the document. In our approach, advertisements are derived from the DTD, since the DTD contains all possible paths from the root to the leaves appearing in related XML documents. A fragment of a DTD, *personal.dtd*, is shown in the Fig. 2 (a). It is used to define the structure and corresponding content of an XML document. Fig. 2 (c) shows the XML conforming to the DTD from Fig. 2 (a), which contains the actual data, e.g., person id, name (including family name and given name), email, URL and link (superior and subordinate). Here, our discussion focuses on the main building block: elements. Our approach can be easily extended to element attributes and content.

In this paper, we use the common interpretation of an XML document as a tree of nodes and consider each path from the root node to a leaf node. Thus, we decompose each XML document into a set of XML paths and each path is represented as $e = /t_1/t_2/.../t_n$, where t_i is the XML element name. These paths are extracted from the document before the publisher submits the document to the network. Thus, a publication routed in our system is actually an XML path annotated with a *pathId* and *docId*. This is transparent to publishers and subscribers who handle entire XML documents. Publishers submit entire XML documents, commonly referred to as publications, and subscribers submit XPath expressions (XPEs), commonly referred to as subscriptions. We use the terms XPE and subscription interchangeably in the rest of this paper.

We use the abstract XPath expression without //-operators as the format of advertisement in the context of XML/XPath data routing. Note that this is not a restriction of our subscription language. Advertisements are a system internal mechanism, which is not exposed to the application or the user. An advertisement is described as $a = /t_1/t_2/.../t_{n-1}/t_n$, where t_i can be either an element name or a wildcard, and a has the same length as the publication it advertises. All advertisements extracted from *personal.dtd* are listed in Fig. 2 (b). For example, the advertisement, $/personnel/person/name/family$ indicates a path from the root element, $personnel$, to one of the leaves, $family$, appearing in the XML document in Fig. 2 (c).

We call an advertisement a *non-recursive* advertisement if it is extracted from a non-recursive DTD. Advertisement a is an example of non-recursive advertisement. A DTD is recursive if it has some element that is defined in terms of itself, directly or indirectly, e.g., the NITF DTD is recursive. We define an advertisement as a *recursive* advertisement if it has recursive elements defined in a DTD. An advertisement may have multiple recursive parts that appear in sequence or are embedded in each other. We classify the recursive advertisements to three categories as below.

Simple-recursive advertisements: There is only one recursive pattern in the simple-recursive advertisement. It is described as $a = /t_1/t_2/.../t_{i-1}\ (/t_i/.../t_j)^+/.../t_n$, where the $+$ operator declares that elements $t_i, ..., t_j$ must occur one or more times in the advertisement. Note that this is not part of XPath syntax, and advertisements are only used within the system, so the extended syntax has no effect to the clients and ap-

```
<!-- <?xml encoding="US-ASCII"?> -->
<!-- Revision: 60 1.2 data/personal.dtd, docs, xml4j2, xml4j2_0_0_d04 -->
<!ELEMENT personnel (person)+>
<!ELEMENT person (name,email*,url*,link?)>
<!ATTLIST person id ID #REQUIRED>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT name (#PCDATA|family|given)*>
<!ELEMENT email (#PCDATA)>
<!ELEMENT url EMPTY>
<!ATTLIST url href CDATA #REQUIRED>
<!ELEMENT link EMPTY>
<!ATTLIST link
  manager IDREF #IMPLIED
  subordinates IDREFS #IMPLIED>
```

(a)

```
/personnel/person
/personnel/person/name
/personnel/person/name/family
/personnel/person/name/given
/personnel/person/email
/personnel/person/url
/personnel/person/link
```

(b)

```
<?xml version='1.0'?>
<!DOCTYPE personnel SYSTEM 'personal.dtd'>
<!-- msg_banner -->
<personnel>
  <person id="a0">
    <name>level 2<family>level 3</family>
    </name>
    <email>level 2</email>
  </person>
  <person id="a1">
    <name>level 2<given>level 3</given>
    </name>
    <email>level 2</email>
    <url href="defaultCDATA0"/>
    <link subordinates="a1" manager="a0"/>
  </person>
</personnel>
```

(c)

Figure 2. DTD, Corresponding Advertisements and XML Document

plications. In the proposed algorithms, we use $a = a_1(a_2)^+a_3$ to simplify the expression, where a_k $(1 \leq k \leq 3)$ is a non-recursive advertisement.

Series-recursive advertisements: A series-recursive advertisement can include more than one recursive patterns in sequence. For example, the advertisement containing two recursive patterns in sequence can be described as $a = /t_1/t_2/.../t_{i-1}(/t_i/.../t_j)^+/t_{j+1}.../t_{l-1}(/t_l/.../t_o)^+/.../t_n$, or more simplified expression with non-recursive advertisements, $a = a_1(a_2)^+a_3(a_4)^+a_5$.

Embedded-recursive advertisements: In an embedded-recursive advertisement, recursive patterns can be embedded in others. A possible case is $a = /t_1/t_2/.../t_{i-1}(/t_i/.../t_{l-1}(/t_l/.../t_o)^+/.../t_j)^+/.../t_n$, or $a = a_1(a_2(a_3)^+a_4)^+a_5$. The embedded-recursive advertisement could be more complicated.

More types of recursive advertisements can be easily defined based on the above three types of advertisements. The matching algorithms for non-recursive and recursive advertisements are presented in [13].

Covering and Merging In this section, we fucus on advertisement-based subscription routing and its optimization techniques, such as covering and merging, the matching between XPE and XML data has been discussed in [9].

First, a novel data structure, called *subscription tree*, is presented for maintaining subscriptions. The data structure captures the covering relations among subscriptions. It can speed up the publication and subscription matching as well. In covering-based routing, if an arriving subscription is covered by an existing subscription in the routing table, the new subscription is not forwarded to the next-hop broker. One the other hand, if the arriving subscription covers existing subscriptions, before it is forwarded, the broker needs to unsubscribe all the subscriptions that are covered by the new subscription. Therefore, the network traffic is reduced by removing the redundant subscriptions and the routing table in the next-hop broker is compacted. Subscriptions are maintained in a tree data structure. The idea is to store the subscriptions according to the covering relationship among them. A subscription at a node in the tree covers all subscriptions in its subtree. Since a covering relation defines a partial order among subscriptions, a tree data structure cannot capture all the covering relations. A subscription node can have only one parent in the tree, but it may be covered by several subscriptions. We allow each node having a set of *super pointers*, which indicate the covering relations with nodes outside its subtree, as shown in Fig. 3. *Super pointers* are shortcuts to subscriptions that the node covers. With super pointers, a node covers its subtree, the nodes with subtrees pointed to by its super pointers, and the nodes with subtrees pointed to by its offsprings' super pointers. The maintenance of the tree and existing super pointers while inserting, and optimizations for subscription searching and insertion are described in detail in [13].

Second, we describe how to determine the covering relationship between two given subscriptions. The covering relation between subscriptions is the containment problem in the context of XPEs. It has been proven that containment of simple path expressions can be tested in PTIME [15]. It is studied as a part of the problem that checking/finding a prefix replacement for a simple query is in PTIME. We detect covering relations of XPEs containing wildcard, /- and //-operator in PTIME, and present covering rules for determining covering relation between single path XPEs. The covering rules used in our algorithms are as follows. We say Sub_1 containing an element t_i covers Sub_2 containing an element m_i at the cor-

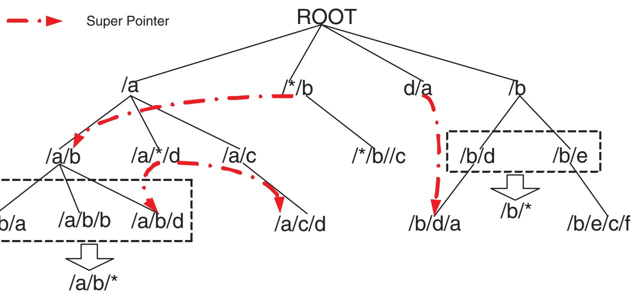

Figure 3. Subscription Tree

responding position, if t_i is a wildcard no matter what m_i is, or $t_i = m_i$, where neither t_i nor m_i are wildcards. The algorithms for detecting covering relation among absolute XPEs, relative XPEs and XPEs with descendant operators are presented in [13].

Third, we exploit the merging rules for XPEs. If there is no covering relation among a set of subscriptions, subscriptions can be merged into a new subscription to create a more concise routing table. In the subscription tree, child nodes of the same parent have a better chance to be merged. As shown in Fig. 3, node $/a/b/a$, $/a/b/b$ and $/a/b/d$ can be merged and they are represented by a new node $/a/b/*$ which is a union of the original XPEs. There was a super pointer pointing to node $/a/b/d$ before merging. This pointer should be removed because there is no covering relations between the pointer owner and the merger. If two nodes are merged, their subtrees become siblings of the merger. For example in Fig. 3, after $/b/d$ and $/b/e$ are merged to $/b/*$, there children are the new node's children. The super pointer at $/b/d/a$ was not changed. To perform the merging in the subscription tree, we define several merging rules (for details see [13].) The idea of these merging rules is to generate a more general operator (e.g., //-operator) or an element (e.g., wildcard) in the merger. Therefore, a more compact routing table is created since many subscriptions with common parts are merged to this merger.

5. Evaluation

We experimentally evaluate the performance of our routing, covering and merging algorithms. We use two different DTDs: the NITF (News Industry Text Format) DTD and the PSD (Protein Sequence Database) DTD to generate data. First, we evaluate the effect of the covering optimization on the routing table size. For the data set with higher degree of overlap, the routing table size is reduced significantly due to covering. Consequently, the routing times at each broker are improved by up to 85% in the most favorable cases because of a more compact routing table, which is generated by removing redundant XPEs due to covering. We also investigate the impact of advertisement-based routing and covering techniques on network traffic. Our experiments suggest that advertisements can be used to avoid flooding of XPath queries, and thus reduce the overall network traffic. In our experiments by up to 76%. Moreover, removing redundant queries by exploring covering relations among subscriptions reduces the network traffic and saves system resources.

6. Conclusion

In this paper, we have studied the problem of efficiently routing XML data through a data dissemination network comprised of an overlay network of content-based routers. In the dissemination network, publishers' DTD files are transformed into advertisements expressed using XPath-like expressions. An advertisement creates a spanning tree rooted at the publisher. Subscribers specify their XPath filters which are forwarded along the reverse paths of *intersecting* advertisements, i.e., those may have potentially interesting XML documents. XML documents from publishers are forwarded along the reverse paths of *matching* XPEs to interested subscribers. Our contributions are: first, we discuss the advertisement-based routing protocol for XML-based data dissemination networks. Advertisements are generated from a DTD file and can be applied for all related XML documents. We discuss both recursive and non-recursive advertisements extracted from DTD files. Moreover, we propose a set of algorithms for the matching of advertisement and XPE. Second, we present some algorithms to determine the covering relations between to arbitrary XPEs, and a novel data structure called subscription tree to maintain the XPEs in an XML router, which maintains the covering relationship among XPEs. The routing time at each broker are improved dramatically because the covering relationship are fully explored among XML queries to reduce the routing table size. Third, we also propose some rules to merge similar XPEs in order to further reduce the routing table size. The evaluation results suggest that the scalability of the system has been improved by applying advertisement-based routing, covering, and merging techniques.

References

[1] N. Bruno, L. Gravano, and N. Doudas. Navigation-vs. index-based XML multi-query processing. In *ICDE*, 2003.
[2] A. Carzaniga, M. J. Rutherford, and A. L. Wolf. A routing scheme for content-based networking. In *INFOCOM*, 2004.
[3] C. Chan, P. Felber, and M. Garofalakis. Efficient filtering of XML documents with XPath expressions. In *ICDE*, 2002.
[4] G. Cugola, E. D. Nitto, and A. Fuggetta. The JEDI event-based infrastructrue and its application to the development of the OPSS WFMS. *IEEE TSE*, 2001.
[5] 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.*, 2003.
[6] Y. Diao, S. Rizvi, and M. Franklin. Towards an internet-scale XML dissemination service. In *VLDB*, 2004.
[7] E. Fidler, H.-A. Jacobsen, G. Li, and S. Mankovski. The PADRES distributed publish/subscribe system. *ICFI'05, UK*.
[8] L. Galanis, Y. Wang, S. Je, and E. DeWitt. Locating data sources in large distributed systems. In *VLDB*, 2003.
[9] S. Hou and H.-A. Jacobsen. Predicate-based filtering of XPath expressions. In *ICDE*, 2006.
[10] G. Koloniari and E. Pitoura. Content-based routing of path queries in peer-to-peer systems. *EDBT*, 2004.
[11] N. Koudas, M. Rabinovich, and D. Srivastava. Routing XML queries. *ICDE*, 2004.
[12] G. Li, S. Hou, and H.-A. Jacobsen. A unified approach to routing, covering and merging in publish/subscribe systems based on modified binary decision diagrams. *ICDCS*, 2005.
[13] G. Li, S. Hou, and H.-A. Jacobsen. Routing of XML and XPath queries in data dissenmination networks. In *Technical Report, U. of Toronto*, Oct,2006.
[14] G. Li and H.-A. Jacobsen. Composite subscriptions in content-based publish/subscribe systems. In *Middleware'05,France*.
[15] T. Milo and D. Suciu. Index structures for path expressions. In *ICDT*, 1999.
[16] G. Mühl. Large-scale content-based publish/subscribe systems. *Ph.D Dissertation*, University of Darmstadt, 2002.