

Sonnet: An Efficient Distributed Content-based Dissemination Broker

Aoying Zhou[†], Weining Qian[‡], Xueqing Gong[†], and Minqi Zhou[†]

[†] Department of Computer Science and Engineering, Fudan University, Shanghai, China

[‡] Software Engineering Institute, East China Normal University, Shanghai, China

[†]{ayzhou, gongxq, zhousinqi}@fudan.edu.cn, [‡]wnqian@sei.ecnu.edu.cn

ABSTRACT

In this demonstration, we present a prototype content-based dissemination broker, called Sonnet, which is built upon structured overlay network. It combines approximate filtering of XML packets with routing in the overlay network. Deliberate optimization technologies are implemented. The running and tracing of the system in a real-life application are to be demonstrated.

Categories and Subject Descriptors

H.2.4 [Database Management]: Systems—*distributed databases, query processing*; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval—*information filtering*; H.3.4 [Information Storage and Retrieval]: Systems and Software—*distributed systems, selective dissemination of information-SDI*

General Terms

Design

Keywords

Distributed publish/subscribe, XML data dissemination, approximate filtering, path digest

1. INTRODUCTION

Publish/subscribe systems have been becoming important applications over the Internet in the past several years. Tens of thousands sources emerge in a short period of time, and even more readers may be involved as data consumers. A common approach for representing such data is to use XML, or more specifically, RSS, to describe the published data, and with the data representation user can express their subscription requests by aggregation.

In this demonstration, we present the Sonnet, a distributed system for content-based dissemination of XML data. Sonnet is featured by the following points. First, data to be disseminated may come from many different sources. Furthermore, the subscribers are supposed to be much more than the publishers in quantity. The subscription request is *content-based*. Therefore, queries over XML documents are supported by the system. Last but not the least, the

```
<?xml version="1.0"?>
<rss version="2.0">
  <channel>
    <item>
      <title>SIGMOD 2005 research paper</title>
      <link>http://doi.acm.org/10.1145/1066159</link>
      <description>
        <a>T. Johnson</a>
        <a>S. Muthukrishnan</a>
        <a>I. Rozenbaum</a>
      <title>Sampling Algorithms in a Stream Operator</title>
      <inproceedings>SIGMOD 2005</inproceedings>
    </description>
  </item>
</channel>
</rss>
```

Figure 1: An example RSS feed.

dissemination process is elegantly integrated into the underlying routing process. It yields the efficiency and scalability of the system.

Some existing systems support keyword-based subscription. However, user may require more powerful querying tools. Taking the RSS feed in Figure 1 as an example. A possible subscription request may be a query to ask for links to papers written by a specific author, such as: ‘\$RSS/channel/item/link[description/author=“T. Johnson”]’. The Sonnet system is designed to support such a kind of subscription over network environment.

The previous work that is most close to our system is distributed XML filtering and dissemination, such as ONYX [1]. ONYX shares the same motivation as Sonnet. One key difference is that ONYX directly use XPath queries as entries in the routing table, thus each forwarding process is essentially a filtering process. However, Sonnet uses an additional header, which is a 128-bit string, for each packet. The header is checked via bit-based operators during the routing process, and this is much cheaper than XML filtering. The routing serves as an approximate dissemination process as well in Sonnet. Thus it achieves better efficiency and scalability of the system.

Sonnet distinguishes itself from the existing content-based dissemination systems by the following three features. First, nodes sharing *similar* dissemination tasks can share feeds with each other. Second, data dissemination is embedded into data routing in the overlay network. Third, optimization techniques, including workload balancing and locality-aware dissemination, are proposed and implemented.

2. SYSTEM OVERVIEW

Sonnet adopts a two-layer structure, which is shown in Figure 2. The lower layer is a routing module. It is responsible for receiving the data packet and forwarding the packet by checking packet header and the entries in the fin-

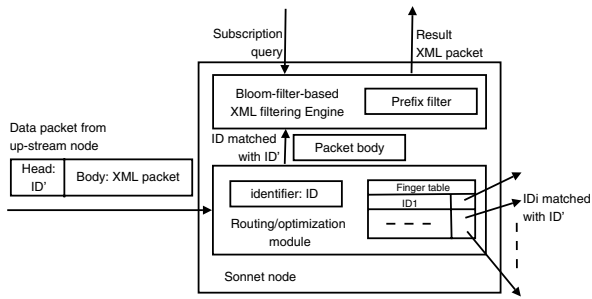


Figure 2: The architecture of a Sonnet node.

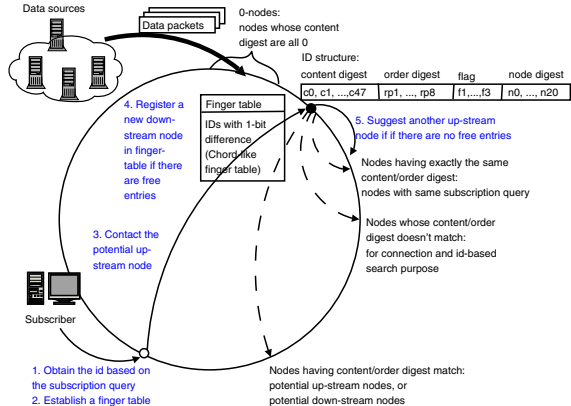


Figure 3: The Sonnet network structure.

ger table pointing to remote nodes. The packet header is a summarization of the content of the packet. The identifier of a node is actually a summarization of subscription request registered on it. Thus, the process of matching the packet header with entries in the finger table is essentially an approximate dissemination procedure. The upper layer is a conventional XML data filtering engine, namely Bloom-filter-based engine [2]. It is responsible for checking if a packet can exactly satisfy the user's subscription request.

The overall Sonnet network structure is shown in Figure 3. The Sonnet overlay network is organized as a ring. Each node has a 128-bit identifier, which consists of two parts: query digest and node digest. Query digest is the summarization of the subscription request registered on the node, while node digest is the hashed value of the node information, including IP address and hardware details. For a node having more than one subscription requests, there is one virtual node for each request. The details about the method for obtaining query digests is introduced in the next section. The basic actions a node may take are:

Subscription or unsubscription Each node may accept subscription request. Currently, linear path expression with predicates is supported. A node identifier is obtained based on the query and the node's information. Then, the node joins the network via the overlay network protocol. The upper-layer filtering module is also constructed. Consequently, the *up-stream node* (i.e. the node which directly provides data) is found, and the subscription is registered (as shown in Figure 3). The unsubscription process is the reverse.

Dissemination When a packet is received, its header is checked with the node identifier. If a match is found, the packet is returned to the filtering module. The packet that can answer the subscription query is returned to the user. Each entry in the finger table is also checked, for those entries matched with the header, the packet is forwarded to the corresponding node.

Optimization and stabilization Even the subscription request is not changed, a node may update its neighboring nodes, i.e., entries in the finger table, for performance reasons. When a neighboring node is found to be unavailable, a stabilization procedure is invoked to maintain the connection of the network.

3. IMPLEMENTATION

The major implementation issues are introduced in this section, while the details could be found in [3].

3.1 Summarization of Subscriptions and Nodes

A query digest consists of three parts: content digest, order digest and flags. A content digest is a 48-bit bit-string. Each tag or predicate in the query is hashed to the bit-string independently. The content digest is the bit-wise OR of the bit-strings of all tags.

Content digest does not contain any information about the orders of the elements and predicates. Order digest is responsible for summarizing this kind of information. First, the path is summarized using the alphabetic order of the elements, where the predicates are put into the end of the path. Then, the *rank of permutation* (RP) is computed. For a complete path, RPs of all prefixes are computed and stored in binary form as *order digest*. 56-bit order digest is used in implementation.

Each XML packet transmitted in Sonnet is attached with a header, which is the path digest of all paths included in the packet. The header is used for routing packets. The routing procedure also serves as an approximate dissemination process, but with no XML document parsing or exact filtering is needed. And it is fast since only bit-string operators are used. This is one of the main advantages of Sonnet over existing content-based subscription systems.

The remaining bits in a node identifier is for node digest. Node digest is a hashed result of the profile about the computer. Therefore, for any two nodes with common subscription query, they may have different node identifiers with high probability.

3.2 Basic Routing

The entries in a Sonnet node's finger table can be classified into three categories. *Content links* point to up-stream or down-stream nodes. *In-group links* point to nodes with exactly the same subscription request. *Maintenance links* point to nodes that don't share subscription interests.

On receiving a packet, its header is retrieved. Each entry in the header p is compared with each entry e in the finger table of node n . A packet is forwarded to a node pointed by the entry if and only if the following three conditions are satisfied: 1) The bits set to '1' in content digest of e are all set to '1' in p 's content digest. 2) The content digest of e is just the same as that of n or has one more bit set to '1' in that of n . And 3) Let l be the number denoted by the flags, rp_1, rp_2, \dots, rp_l of e should be the same as those of p .

Besides the down-stream nodes, a packet should be transmitted to the nodes with exactly the same subscription query. This kind of nodes are discovered via the following three rules: 1) The bits set to '1' in content digest of e are exactly the same as those set in n . 2) The order digest rp_1, rp_2, \dots, rp_l of e are exactly the same as those in n . And 3) The l denoted by $flags$ of e is the same as that in n .

Thus, these nodes form a small group, and the packet is multi-casted to all members in it. For packets forwarded by a group member, they will not be forwarded to down-stream nodes, but to other group members further.

This basic routing scheme guarantees that all nodes can receive packets satisfying the subscription query[3].

3.3 Optimization

In the basic routing scheme, computation cost is distributed to nodes along the transmitting path in the network, while the network bandwidth consumption is distributed to different links between up-stream/down-stream pairs. One remaining important issue is workload balance. Since the contents of packets and subscription are not evenly distributed, randomization provided by hash functions is not sufficient for distributing the workload to nodes evenly.

3.3.1 0-node utilization

The nodes whose content digest part are all zero are called *0-nodes*. When a new packet is generated by a source, the source randomly generates a node digest. Then, the packet is sent to a 0-node corresponding to that node digest. The 0-node is responsible for forwarding the packet to nodes subscribing its content.

0-nodes provide a mechanism to distribute the workload when the packet enters the Sonnet network. Therefore, no links for a pair of up-stream/down-stream nodes would have higher workload over other links connecting two nodes from the same groups respectively.

3.3.2 Node degradation

Another kind of workload unbalance problem appears when the number of nodes in up-stream is much less than that in down-stream. In such cases, the up-stream nodes are overloaded, thus they will become the bottleneck in the network.

The number of nodes in a group is estimated. When it is found that up-stream nodes are suffering overloading problem, the node degradation process is invoked. The degraded node cuts the last element in its subscription query and modifies the finger table according to its new query. The degraded node has to spend more network bandwidth and computation power for routing and filtering packets.

When the nodes in a group are no longer overloaded, the degraded nodes are upgraded via a reverse procedure of degradation. This flexible scheme provides a workload balancing method for the dynamic network environment.

3.3.3 QoS-based node selection

The workload is not the only issue in performance. A node may tend to choose a more stable and efficient up-stream node.

We use a QoS-based scoring mechanism. Each up-stream node is scored by its down-stream nodes. When a node is choosing its up-stream nodes, it can first query the QoS scores, and then select the most appropriate one. This solution enables further optimization for different applications with corresponding scoring schemes can be defined.

3.4 Experimental Result Summary

Sonnet is implemented by using Java. Simulation experiments on 500 to 3000 nodes are conducted [3]. The summarization of the conclusions are as follows.

- The 128-bit identifier is powerful for approximate data filtering, which keeps the false positive rate under 20%.
- The degradation technique can dynamically adjust the computation and networking resource for processing the heaviest workload in the network.
- Sonnet outperforms other packet transmitting protocols in terms of workload balancing and reliability
- The more nodes (subscribers) participate in the network, the better the performance is, which is an intrinsic advantage of peer-to-peer systems.

4. DEMONSTRATION OUTLINE

In this demonstration, a network with at least four physical nodes and tens of virtual nodes will be established. Following features of Sonnet will be demonstrated.

1. XML feeds are simulated for publishing DBLP publication list, CiteSeer publication information in OAI format, and/or DBWorld messages annotated by XML tags.
2. We will show how to do content-based subscription without specify the source. This is performed by writing restricted XPath queries. We will show that users can obtain the published messages accurately in this purely decentralized environment.
3. We will illustrate the dissemination process and mechanism of Sonnet as well. The tracking of the forwarding path of the packets could be visualized to demonstrate how Sonnet distributes the dissemination subtasks to the nodes, and how the computation of one node is shared by others.
4. We will demonstrate how Sonnet responds to a dynamic environment with node's joining/leaving and workload changing. The change of the network organization will be displayed, and the performance measurements are to be displayed as well. We will show how a node acts while a measurement changes, and how the action affects the performance.

5. REFERENCES

- [1] Y. Diao, S. Rizvi, and M. Franklin. Towards an internet-scale xml dissemination service. In *Proceedings of the 30th VLDB Conference*, 2004.
- [2] X. Gong, W. Qian, Y. Yan, and A. Zhou. Bloom filter-based xml packets filtering for millions of path queries. In *Proc. of the 21st ICDE Conference*, 2005.
- [3] W. Qian, L. Xu, A. Zhou, and M. Zhou. Sonnet: Subscription using path queries over structured overlay networks. *To appear in Frontiers of Computer Science in China*, 1(2), 2007.