P2PENS: Content-Based Publish-Subscribe over Peer-to-Peer Network

Tao Xue, Boqin Feng, and Zhigang Zhang

School of Electronics and Information Engineering,
Xi'an Jiao Tong University,
710049 Xi'an, China
Xt73@163.com

Abstract. Content-based publish/subscribe systems have recently received an increasing attention. Efficient routing algorithms and self-organization are two challenges in this research area. Peer-to-peer network topologies can offer inherently bounded delivery depth, load sharing and self organization. In this paper, we present a content-based publish/subscribe system built on top of a dynamic peer-to-peer overlay network. A scalable routing algorithm using Pastry network is presented that avoids global broadcasts by creating rendezvous nodes. Self-organization and fault-tolerance mechanisms that can cope with nodes and links failures are integrated with the routing algorithm resulting in a scalable and robust system. The experiment results reveal that our system has better routing efficiency, lower cost and less routing table size at the event brokers.

1 Introduction

Content-based publish/subscribe (pub/sub) middleware differs significantly from traditional subject-based pub/sub system, in that messages are routed on the basis of their content rather than pre-defined subjects or channels. The flexibility of content-based systems comes at the expense of added challenges in design and implementation. The open issue is how to efficiently route events to subscribers interested in them through the overlay network of event brokers. The next challenge is dynamic reconfiguration of the topology of the distributed dispatching infrastructure. That is, the overlay network should dynamically adapt to the changing of lower network topology raised by node or link failure, requiring no manual tuning or system administration. Some approaches for content-based pub/sub [1] [2] [3] [4] [5] [6] have been presented, but to the best of our knowledge, none of them provides any special mechanism to support the self-organization and fault-tolerance addressed by this paper.

On the other hand, peer-to-peer (P2P) systems have recently received significant attention in both academia and industry. P2P systems are distributed systems without any centralized control or hierarchical organization, where the software running at each node is equivalent in functionality. Recent work on peer-to-peer overlay networks offers a scalable, self-organizing, fault-tolerant substrate

H. Jin, Y. Pan, N. Xiao, and J. Sun (Eds.): GCC 2004, LNCS 3251, pp. 583–590, 2004.

for decentralized distributed applications such as CAN [7], Chord [8], Pastry [9] and Tapestry [10]. Essentially, these protocol all adopt DHT(Distributed Hash Table)strategy, which provides a mapping of keys in some keyspace to machines in the network and a lookup protocol to allow any searcher to find the particular machine responsible for any key.

This paper presents a content-based publish/subscribe middleware P2PENS that combines the P2P system and content-based publish/subscribe system. In the bottom layer, Pastry is used to realize an overlay network of arbitrary topological structure; in the top layer, a content-based routing protocol with fault-tolerance and self-organizing mechanism is used to construct event dispatcher trees. Furthermore, dis-patcher trees will be reconstructed automatically, which vary with the interests of subscribers.

2 System Architecture

A novel pub/sub architecture we present is illustrated in Fig. 1. Pastry is selected as lower layer communication mechanism in the event broker network. Every event broker will run two protocols. One is Pastry's P2P protocol that handles join or leaving of nodes and routes messages to a rendezvous point. The other protocol is our content-based routing protocol that constructs event dispatcher trees and forwards events along the trees. Event brokers are interconnected with arbitrary topological structure, and communicate in point-to-point mode.

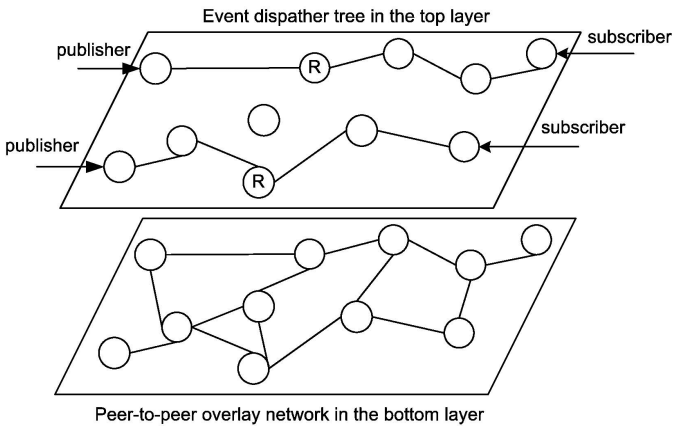

Fig. 1. System Architecture

Unlike the state of the art content-based publish/subscribe systems, which have only one event dispatcher tree for all of events, our system borrows rendezvous point mechanism from Herald [11] and uses it to constructs multi-dispatcher trees. In our system, events are classified by event type. Every event

type corresponds to a rendezvous point in the network. Rendezvous point is a special event broker; it is the root of an event dispatcher tree. Events and subscriptions firstly meet in their corresponding rendezvous point. Then events are filtered and propagated along the dispatcher tree stemmed from the rendezvous point. Fig.1 illustrates two dispatcher trees that represent two different event types respectively. The rendezvous point identified with R. Clearly the mapping between an event type and its rendezvous point is a key problem we must resolve, which will be discussed in following section.

3 Routing Algorithm

The design of routing algorithm plays a decisive role for the scalability of pub/sub middleware. The routing algorithm presented here is suited to overlay networks with arbitrary topology, and need not globally broadcast events or subscriptions. Furthermore, it featured in self-organization and fault-tolerance, which distinguished it from existing routing algorithms.

3.1 Rendezvous Point

The purpose of introduction of rendezvous points is to have related events and subscriptions met and construct a dispatcher tree. There can be a lot of rendezvous points in the network; each rendezvous point takes charge of one event type. Similar to Core-Based Multicast Trees [12], multi-rendezvous points share network flow and the burden of matching events against subscriptions, which increases the scalability largely. We use Rt to indicate the rendezvous point of event type t.

Obviously, we should map event types to rendezvous points. Our system uses Pastry P2P network at the bottom layer to set up or search rendezvous point of an event type. All Events published or subscriptions subscribed contain an event type t. The result of Pastry Hash computing on t is the key of the event or subscription. Pastry will route the event or subscription to the node with the nodeId that is numerically closest to the key. This node is the rendezvous point of event type t, namely Rt.

3.2 Propagation of Events and Subscriptions

A subscriber sends its subscriptions to a connected event broker. Every event broker got subscriptions works according to the following steps:

1: Extract event type t from the subscription and perform Hash computing on t to get the key.
2: Save the subscription and the nodeId of the node who sent this subscription to routing table.
3: Determine if this event broker is Rt, if not, call Pastry's routing operation to route the subscription to the next hop according to the key.

A publisher sends events to a connected event broker. Every event broker got events works according to the following steps:

1: Determine whether the event has past Rt by checking the event's Rt field, if yes, then go to step 4.
2: Extract the event type t from the event and perform Hash computing on t to get the key.
3: Determine if this event broker is Rt, if not, then go to step 5; otherwise set the event's Rt field.
4: Match the event against subscriptions in routing table, and forward the event along reverse propagation path of every subscription matched successfully, then the algorithm stop.
5: Call Pastry's routing operation to route the event to the next hop according to the key, then the algorithm stop.

Events and subscriptions of the same type meet at their rendezvous point, and then events are forwarded along reverse propagation path of each subscription matched successfully. Fig. 2 shows the algorithmic process. Publisher P1 publishes event e1 and subscriber S1, S2 subscribe subscription s1 and s2 respectively. Without loss of generality, suppose the event type of e1 and that of s1, s2 are the same and they are matched. Firstly, in term of subscriptions propagation algorithm, s1 and s2 are routed to the rendezvous point Rt(node 4 in Fig. 2) and their propagation path are also recorded by every node along the path. Then e1 is routed to the rendezvous point Rt according to event propagation algorithm. Rt matches e1 against s1and s2, if successful, e1 is forwarded along the reverse propagation path of s1 and s2 until e1 gets to s1 and s2 respectively.

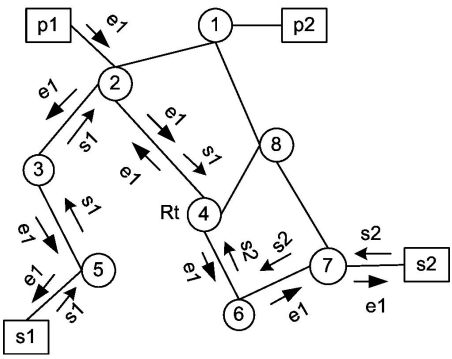

Fig. 2. Pastry-based Routing Algorithm

3.3 Optimization of Routing Algorithm

The main problem of the routing algorithm described above is that all of the events and subscriptions with the same event type must firstly reach their rendezvous point, which may causes the rendezvous point overload in some cases.

With the introduction of advertisement and coverage relation between subscriptions [4] [5], our algorithm can be further optimized.

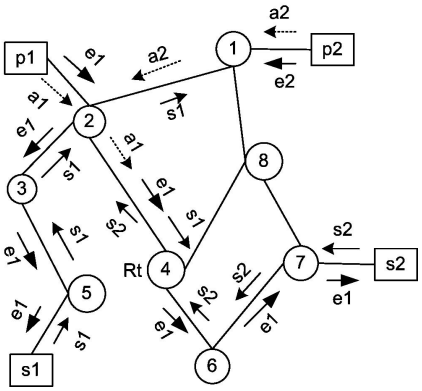

Fig. 3. Pastry-based Routing Algorithm with Advertisement

Fig. 3 illustrates an example of routing optimization. Suppose that event e1, e2, subscription s1, s2 and advertisement a1, a2 have the same event type. E1 matches s1 and s2, but e2 and s1, s2 are mismatched. Moreover a1 covers a2 and s1 covers s2. Firstly, a1 and a2 are sent by P1 and P2 respectively. They will be routed to their rendezvous point, but a2 will not be forwarded at node 2 because a1 covers a2, so only a1 will reach the rendezvous point at last. S1 is routed according to the subscription propagation algorithm. When it reaches node 2, due to advertisement a1 and a2 are the same event type with s1, so s1 is forwarded along reverse propagation path of advertisement (s1 is forwarded to node 1 in Fig. 3) on the one hand. On the other hand, s1 is routed to the rendezvous point. S2 is also routed to the rendezvous point and continue to be forwarded along reverse propagation path of a1. At node2 because s1 covers s2 and s1 has been forwarded to node 1, so the propagation of s2 will be stopped. Then e1 and e2 are published; they are forwarded along reverse propagation path of subscriptions. When e2 reaches node 1, it's discarded because it's not matched with s1. When e1 reaches node 2, since e1 matches both s1 and s2, so e1 is forwarded along the reverse propagation path of s1 and s2 until e1 gets to s1 and s2 respectively.

From the illustration above it's clear that, due to the introduction of advertisement, the propagation of mismatched events are restricted (e.g. e2 is discarded at node 1 in Fig. 3), and events need not be definitely routed to rendezvous points at first (e.g. e1 is forwarded from node 2 to the direction of S1 source in Fig. 3). Furthermore, coverage relation reduces unnecessary propagation of subscriptions and advertisements (e.g. node 2 only forwards a1 and s1 in Fig. 3).

3.4 Self-organizing and Fault-Tolerance

Self-organizing and fault-tolerance is very important for a large-scale middleware system with many nodes where link or node failures are frequent. Our system makes use of the fault-tolerance mechanisms provided by Pastry layer that handles join and leaving of the nodes. Besides, a soft state mechanism and a heartbeat protocol is developed, which ensure that unused routing paths can be automatically deleted and the neighbors of an event broker are reachable and alive. Our system can survive multiple link and broker failures and to adapt it's routing state so that it can still deliver events to subscribers.

In the event dispatcher trees rooted from rendezvous points (namely, reverse propagation path of every subscription), each node sends heartbeat message to its adjacent children periodically. A child suspects that its parent is faulty when it fails to receive heartbeat messages. Moreover, the entire state (including subscriptions and advertisements) in the event brokers is soft state that has to be refreshed periodically. We associate a time contract with the state. If the child who sent subscriptions or advertisements does not explicitly refresh the time contract, these subscriptions or advertisements associated with it is reclaimed. Thus, outdated routing will disappear when not explicitly refreshed.

When a child in a dispatcher tree detects that its parent has failed, it calls Pastry to route subscriptions or advertisements to a new parent, thus repairing the dispatcher tree. Original invalid routing path will be removed automatically because of expiration of time contract.

4 Evaluation

We implement a prototype simulation system that is based on the FreePastry platform [13]. It can be further classified into non-optimization version P2PENS and advertisement version P2PENSadv. Due to space limitation, only parts of experiment results are shown below.

(1)Average event delay. It is calculated by dividing the time spent by the total number of events received. As shown in Fig. 4, in all three cases, the average latency decreases and tends to reach a constant value as more subscribers are added to the system. The reason of the phenomenon is likely that subscribers converge in every event broker and the event dissemination tree becomes more densely populated. Moreover, the latency of Siena is the largest and it's sensitive to the number of subscribers. Clearly, P2PENSadv gains the advantage over other algorithms. Compared with P2PENS, it takes advantage of advertisement message to optimize routing, through which not all of events need to be routed to the rendezvous point firstly.

(2)Total cost. It is calculated by summing the costs of all node-to-node message traffic. As shown in Fig. 5, Siena's total cost increases very quickly because the hierarchical topology and routing algorithm adopted by Siena is rather statically configured and all of events must be forwarded to the root node whether it's necessary or not. On the contrary, total cost of P2PENS and P2PENSadv are less than that of Siena as different event type has different dispatcher tree.

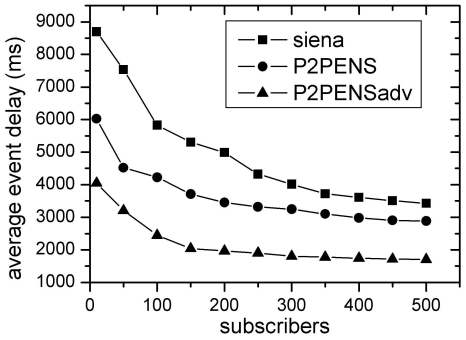

Fig. 4. Comparing in average event delay metric

Especially, the total cost of P2PENSadv is the least because P2PENSadv pulls subscriptions closely to publishers using advertisement message, filtering as early as possible, which avoids unnecessary propagation of events.

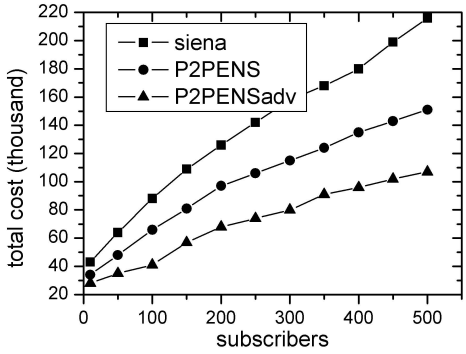

Fig. 5. Comparing in total cost metric

5 Conclusion

Publish/subscribe middleware is an important building block for distributed applications over the Internet. Scalability and fault-tolerance is two big challenges for the design of the middleware. In this paper we present a content-based publish/subscribe middleware that uses a Pastry overlay network as its bottom infrastructure. Its content-based routing algorithm can dynamically adapt to the changing of lower network topology raised by node or link failure, requiring no manual tuning or system administration. In addition, our system has no restrict on the overlay network topology, which further increases the robustness and flexibility of the middleware.

References

1. B. Segall, D. Arnold, J. Boot, M. Henderson, and T. Phelps. Content based routing with elvin4. In Proceedings AUUG2K, Canberra, Australia, June 2000.
2. IBM. Gryphon: Publish/subscribe over public networks. Technical report, IBM T. J. Watson Research Center, 2001.
3. Banavar. G, Chandra. T, Mukherjee. B, Nagarajarao. J, Strom. R, Sturman. D. An Efficient Multicast Protocol for Content-based Publish-Subscribe Systems. Proceedings of IEEE International Conference on Distributed Computing Systems 99, Austin, TX, 1999.
4. A. Carzaniga, D. S. Rosenblum, and A. L. Wolf. Design and evaluation of a wide-area event notification service. ACM Transactions on Computer Systems, 19(3):332-383, 2001.
5. A. Carzaniga and A. L. Wolf. Content-based networking: A new communication infrastructure. In NSF Workshop on an Infrastructure for Mobile and Wireless Systems, Scottsdale, AZ, Oct. 2001.
6. G. Cugola, E. DiNitto, and A. Fuggetta. The JEDI event-based infrastructure and its application to the development of the OPSS WFMS. IEEE Transactions on Software Engineering, 27(9), 2001.
7. S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In Proc. SIGCOMM (2001)
8. I. Stoica, R. Morris, D. Karger, Kaashoek, H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proc. SIGCOMM (2001)
9. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In Int. Conf. on Distributed Systems Platforms (Middleware), Nov. 2001, LNCS 2218, pp. 329-350.
10. B. Zhao, J. Kubiatowicz, and A. Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Tech. Rep. UCB/CSD-01-1141, Computer Science Division, U. C. Berkeley, Apr. 2001.
11. L. F. Cabrera, M. B. Jones, and M. Theimer. Herald: Achieving a Global Event Notification Service. In Proc. of the 8th Workshop on Hot Topics in Operating Systems (HotOS-VIII), 2001.
12. T. Ballardie, P. Francis, and J. Crowcroft. Core Based Trees (CBT). In ACM SIGCOMM 93, Ithaca, N.Y., USA, 1993.
13. Rice University, Houston, USA. Pastry project. http://freepastry.rice.edu/.