An Optimized Peer-to-Peer Overlay Network for Service Discovery

+Institute for Infocomm Research, A*STAR, 21 Heng Mui Keng Terrace, Singapore 119613
+Department of Electrical & Computer Engineering, National University of Singapore
4 Engineering Drive 3, Singapore 117576
E-mail: {guohq, daqing, lhn, lwong, szheng, ykkoh}@i2r.a-star.edu.sg

Abstract

In this paper, we propose DINPeer, an optimized peer-to-peer (P2P) overlay network for service discovery by overcoming limitations in current multicast discovery approaches and P2P overlay systems. DINPeer exploits a spiral-ring method to discover an inner ring with most powerful nodes (DIN Nodes) to form a logical DINloop. With the facilitation of the DINloop, multiple DIN Nodes easily form Steiner trees using Steiner tree-based heuristic routing algorithm. DINPeer further integrates the DINloop and Steiner trees with the P2P overlay network. The key features of DINPeer include that multiple DIN Nodes function as the Rendezvous Points (RPs) for theirs associated logical spaces respectively, and Steiner trees facilitate the communication among multiple DIN Nodes. Multiple powerful DIN Nodes release the burden on the centralized server and the self-recovered DINloop avoids the single point of failure. Simulations show that DINPeer is able to reduce multicast delay for the fast service discovery.

1. Introduction

The decreasing cost of networking technology and network-enabled devices is enabling the large scale deployment of such networks and devices so as to offer many new and innovative services to users. For example, when you carry with your mobile computer to enter a room, or drive on the road, various services have been made available. On a small scale, your mobile computer may want to find a local printer to print a file. On a larger scale, your mobile computer may want to know about the traffic situation from all traffic-cameras along I-280 between Palo Alto and San Francisco. Thus, among hundreds of thousands of services available in the environments, the key challenge is to develop a powerful and robust mechanism to discover the appropriate service for a given task. Currently, there are two basic ways to look up a service: via unicast discovery and via multicast discovery [1]. With unicast discovery, a client must discover a lookup server first, and after having obtained a proxy to the lookup server, the client can now query it to discover the desirable services. Unlike unicast discovery, multicast discovery is a very powerful mechanism, because it does not need to know where a lookup server or desired service is running.

In reality, however, the range over which multicast discovery works is dependent on how a network is configured. Multicast discovery will only work across a subnet or small Local Area Network (LAN). In large organizations, it is likely to find that multicast discovery cannot be applied to locate all of the lookup services. This might be because they are running on different subnets [1]. To overcome this, Jini uses a technique known as “federating” lookup servers and uses “tunneling” to address this problem so that it is possible for a client on subnet A to discover and use services running on subnet B [1]. In Service Location Protocol (SLP), for larger networks, one or more Directory Agents are used to function as a cache that contains all the services from Service Agents [2]. However, those approaches have some drawbacks. First, the lookup server/Directory Agent is assigned manually and maybe it is not the most powerful node. Second, the centralized lookup server/Directory Agent can potentially be subjected to single-point of failure and overloading. Third, clients must be prepared for the possibility that the service information they obtain from Directory Agent is stale [2]. Finally, it may face implementation problems because not all routers support network-level multicast. Today many Internet Service providers (ISPs) are still reluctant to provide a wide-area multicast routing service [3].
Because of problems facing the deployment of a centralized service server and network-level multicast, the application-level multicast based on peer-to-peer (P2P) overlay networks has gained in popularity. Recent works on P2P overlay network offer scalability and robustness for the advertisement and discovery of services. Pastry [4], Chord [5], CAN [6] and Tapestry [7] represent typical P2P routing and location schemes. Furthermore, there has been a number of works reported on adding multicast schemes and applications on P2P object platform, e.g., Scribe [8], CAN-Multicast [9] and Bayeux [10]. Compared to native network-level multicast, application-level multicast has a number of advantages. First, a major advantage is that most proposals do not require any special support from network routers and can therefore be deployed universally. Second, the deployment of application-level multicast is easier than network-level multicast by avoiding issues related to inter-domain multicast. Third, the P2P overlay network is fully decentralized. Thus, application-level P2P overlay multicast is very promising for service discovery.

However, application-level P2P overlay multicast also suffers some disadvantages. Due to the fact that the underlying physical topology is hidden, using application-level multicast increases the delay to deliver messages compared to network-level multicast. A node’s neighbors on the overlay network need not be topologically nearby on the underlying physical network. This can lead to inefficient routing because every application-level hop could potentially be between two geographically distant nodes. In addition, the routing in the P2P overlay network does not consider the load on the network. It treats every peer as having the same power. Further, in P2P tree-based multicast, the Rendezvous Point (RP) can potentially be subjected to overloading and single-point of failure.

In this paper, we propose DINPeer, an optimized P2P communication overlay network for multicast service discovery. DINPeer exploits a spiral-ring method to discover an inner ring with most powerful DIN Nodes to form a logical DINloop. With the facilitation of the DINloop, multiple DIN Nodes easily form Steiner trees using Steiner tree-based heuristic routing algorithm. DINPeer further integrates with the P2P overlay network. DINPeer has two key features. First, DINPeer uses the DINloop instead of a RP as multicast sources. Multiple DIN Nodes function as the RPs for each logical space respectively. Second, Steiner trees facilitate the communication among multiple DIN Nodes.

The benefits of DINPeer in service discovery can be summarized as follows.

- In traditional service discovery, a centralized server is easily overwhelmed by large access traffic. DINPeer with multiple powerful DIN Nodes releases the burden on the single centralized server.
- A centralized server can potentially be subjected to single-point of failure. In DINPeer, the DINloop formed by multiple DIN Nodes can be self-recovered.
- The delay is crucial for fast service discovery so that clients can response in time. DINPeer multicast can reduce multicast delay.

The details of DINPeer will be elaborated in following section.

2. Solution Overview

DINPeer is a massively scalable middleware that combines the strengths of multicast and P2P systems. DINPeer exploits a spiral-ring method to discover an inner ring with most powerful DIN Nodes to form a logical DINloop. With the facilitation of the DINloop, multiple DIN Nodes easily form Steiner trees using Steiner tree-based heuristic routing algorithm. DINPeer further integrates with the P2P overlay network. DINPeer has two key features. First, DINPeer uses the DINloop instead of a RP as multicast sources. Multiple DIN Nodes function as the RPs for each logical space respectively. Second, Steiner trees facilitate the communication among multiple DIN Nodes.

The benefits of DINPeer in service discovery can be summarized as follows.

- In traditional service discovery, a centralized server is easily overwhelmed by large access traffic. DINPeer with multiple powerful DIN Nodes releases the burden on the single centralized server.
- A centralized server can potentially be subjected to single-point of failure. In DINPeer, the DINloop formed by multiple DIN Nodes can be self-recovered.
- The delay is crucial for fast service discovery so that clients can response in time. DINPeer multicast can reduce multicast delay.

The details of DINPeer will be elaborated in following section.

3. Details of DINPeer

This section describes the detail of our solution for multicast service discovery.

3.1. Finding an Optimal DINloop & DIN Nodes

We exploit a spiral-ring method to find an inner ring with most powerful DIN Nodes to form a logical DINloop. First, a big outer ring with all nodes is formed (Fig. 1). A new node is always inserted between two nearest connected nodes when forming the big outer ring.

Second, we use a spiral-ring method to find an inner-most ring with most powerful DIN Nodes (Fig. 1). The nodes in the inner spiral ring must have
higher power than the nodes in the outer ring. Here, the heterogeneity of P2P networks can be exploited by picking nodes with higher computing power. The formation of the inner ring is not limited, as nodes with less power are dropped if enough nodes are available. We may assign the desirable number of nodes in the inner-most ring.

3.2. Forming Steiner Tree & Integrating with P2P Overlay Network

After finding an optimal DINloop and DIN Nodes, multiple DIN Nodes form Steiner trees using the well-known Steiner tree-based heuristic routing algorithm. This Steiner tree is formed using an explicit join method.

Then the DINloop and Steiner tree are integrated with the P2P overlay network [4]. The routing algorithm in DINPeer is similar to the routing algorithm in Pastry P2P system [4], but it integrates with DIN Nodes and Steiner tree. Given a message, the node uses the routing table that is constructed using nodes in its own space and forwards the message to a node locally. If there is no node that can be routed to, the message is marked to differentiate itself from the beginning message and forwarded to its associated DIN Node. Then the DIN Node forwards the marked message to all other DIN Nodes through the Steiner tree. When each DIN Node receives the message, it forwards the message to the node in its own space respectively. Finally, a desirable service is discovered. This routing algorithm can make sure that the routing is done within its own space so as to achieve the location-awareness routing.

3.3. Optimizing Application-Level P2P Overlay Multicast

In the current publish/subscribe system [8], a node is formed as a RP for publishers and subscribers. Subscribers send a message toward the RP via P2P overlay network and registration is recorded along the path. A publisher sends data to the RP and the RP forwards the data along multicast tree formed by the reverse paths from the RP to all subscribers.

In DINPeer, multiple powerful DIN Nodes are used to replace a single RP. Each DIN sub-node finds the nearest DIN Node and becomes a member of a logical space associated to this DIN Node (Fig. 2).
nearest DIN Node forwards the message along other multicast branches to the rest members in its own space, and the nearest DIN Node also forwards the message to the other DIN Nodes along the Steiner tree. One example of DINPeer multicast is shown in Fig. 3. The dotted arrow lines are the paths for the multicast traffic.

3.4. Self-Recovering the DINloop

When a DIN Node decides to leave the DINloop, it sends a quit message to its two adjacent DIN Nodes so that they can set up the connection between them. The nearer one of its two adjacent DIN Nodes is used as the new DIN Node and the quitting DIN Node informs its child nodes in its own space about the new DIN Nodes. Then the child nodes associate to the new DIN Node.

If a DIN Node or a link fails, the DINloop is broken. The nearest DIN Node, which detects the failure, will loop back the message with its nodeId. Another adjacent DIN Node, which also detects the failure, receives the loop-back message and knows the opposite DIN Node. Then it will connect to the opposite side DIN Node again to form a new DINloop (Fig. 4).

4. Experimental results

First, we compare DINPeer multicast with three types of application-level multicast trees, i.e., unidirectional tree, bidirectional tree and shortest path tree. In unidirectional tree multicast (Fig. 5a), a source sends data to the RP (think-dotted arrow lines) and the data cannot be forwarded to the branches before reaching the RP. When the data reaches the RP, the RP forwards the data along the multicast tree to all members (thin-dotted arrow lines). In bidirectional tree multicast (Fig. 5b), a source sends data to the RP and the data can be forwarded to other branches on the way toward the RP. When the data reaches the RP, the RP forwards the data along the other multicast branches to the rest members. We can see the different thin-dotted arrow lines in Fig. 5b. In shortest path tree multicast (Fig. 5c), the source is the root of the tree and the message is forwarded along the shortest path tree to all members.

The simulations ran on the network topologies, which were generated using the Georgia Tech [12] random graph generator according to the transit-stub model [13].

We used the graph generator to generate different network topologies. The number of nodes in the transit domain was changed and the number of nodes in stub domains was fixed, i.e., 6000. We randomly assigned computing power ranging between 1Gbps to 10Gbps to the nodes in the transit domain, used the range of 100Mbps to 500kbps for the nodes in the stub domains. We assumed that all
the overlay nodes were members of a single multicast group. Using the spiral-ring method, the different number of DIN Nodes was obtained. For other application-level multicast, we randomly chose a DIN Node as a RP since all DIN Nodes are powerful and suitable to function as the RP. The results of simulation are shown in Fig. 6.

The legend in Fig. 6 is explained as below.

\[
\begin{align*}
\text{DINPeer} \_ \text{SPT} &= \frac{\text{Average Delay in DINPeer Multicast}}{\text{Average Delay in Shortest Path Tree}} \\
\text{DINPeer} \_ \text{BT} &= \frac{\text{Average Delay in DINPeer Multicast}}{\text{Average Delay in Bidirectional Tree}} \\
\text{DINPeer} \_ \text{UT} &= \frac{\text{Average Delay in DINPeer Multicast}}{\text{Average Delay in Unidirectional Tree}}
\end{align*}
\]

![Multicast Delay Comparison](image_url)

**Figure 6.** Comparison with application-level multicast on the effect of DIN Node number

From Fig. 6, it is clear that when the DIN Node number increases, DINPeer\_SPT, DINPeer\_BT and DINPeer\_UT are decreased. When the DIN Node number is 1, DINPeer\_BT equals to 1 because at this scenario, DINPeer multicast is the same as the bidirectional tree multicast. When the DIN Node number is larger than 1, the average delay in DINPeer multicast is lower than that of unidirectional tree multicast and bidirectional tree multicast, as DINPeer\_UT and DINPeer\_BT are less than 1.

When the DIN Node number is large enough, the multicast delay in DINPeer is close to the optimal multicast delay in shortest path tree multicast since DINPeer\_SPT is close to 1. The reason is that if the DIN Node number equal to the total nodes number, all nodes become DIN Nodes and form the Steiner tree for multicast traffic, then DINPeer multicast is similar to shortest path tree multicast and the multicast delay in DINPeer is close to the multicast delay in shortest path tree multicast. However, shortest path tree requires each source to build up one multicast tree and therefore it is not really implemented by other P2P overlay multicast systems. Most P2P overlay multicast systems apply unidirectional tree multicast. From our simulation results, we can obtain that DINPeer multicast can achieve better performance than other P2P overlay multicast systems.

From Fig. 6, we also obtain that the higher the DIN Node number, the better performance DINPeer multicast can achieve. On the other hand, if there is the limit number of same services, as the number of DIN Node number increases, the probability to find the desirable service in the own space becomes less. Using the routing algorithm in Subsection 3.2, if the desirable service cannot be obtained in the own space, the associated DIN Node will multicast the query message to all other DIN Nodes along the Steiner tree and each DIN Node forwards it to the node in its own space respectively. Thus, this can cause more traffic in the overlay network. Therefore, there is the trade-off to choose the desirable DIN Node number.

Then we compare DINPeer multicast with network-level bidirectional tree multicast and network-level unidirectional tree multicast. The results are shown in Fig. 7.

The legend in Fig. 7 is explained as below.

\[
\begin{align*}
\text{DINPeer} \_ \text{BT}_2 &= \frac{\text{Average Delay in DINPeer Multicast}}{\text{Average Delay in Network Bidirectional Tree}} \\
\text{DINPeer} \_ \text{UT}_2 &= \frac{\text{Average Delay in DINPeer Multicast}}{\text{Average Delay in Network Unidirectional Tree}}
\end{align*}
\]

![Multicast Delay Comparison](image_url)

**Figure 7.** Comparison with network-level multicast on the effect of DIN Node number

From Fig. 7, the average delay in DINPeer multicast is lower than that of network-level unidirectional tree multicast, as DINPeer\_UT2 is less than 1 when the DIN Node number is large. On another hand, other topology-aware routing
techniques are currently able to achieve an average delay stretch (delay penalty) of 1.4 to 2.2, depending on the Internet topology model [14]. The average delay penalty of 1.4 in other P2P overlay network reported in [14] is shown as P2P_1 in Fig. 7. The average delay penalty of 2.2 in other P2P overlay network reported in [14] is shown as P2P_2 in Fig. 7. Since the simulation conditions between our scheme and other P2P systems are different, the difference in quantity cannot be calculated. We only show the general difference. From the results, we obtain that DINPeer has the opportunity to achieve better performance than other P2P overlay multicast systems.

In a brief, from the above results, we can obtain that DINPeer is able to reduce multicast delay for the fast service discovery.

5. Conclusion

Current service discovery based on the centralized server and network-level multicast has some limitations. Due to the problems facing deployment of network-level multicast, application-level P2P overlay multicast has gained in popularity. However, it also suffers from several disadvantages. First, efficient communication in P2P system is a major challenge due to the fact that the underlying physical topology is hidden. Second, every node in the P2P overlay network is equal and the routing does not consider the load of the network. Third, in the P2P tree-based multicast, the RP can potentially be subjected to overload and single-point of failure.

We propose DINPeer, an optimized P2P overlay network for service discovery by overcoming limitations in current multicast service discovery approaches and P2P overlay systems. DINPeer exploits the spiral-ring method to discover an inner ring with most powerful DIN Nodes to form the logical DINloop and multiple DIN Nodes form Steiner trees. DINPeer further integrates the DINloop and Steiner trees with the P2P overlay network. The key features of DINPeer include that multiple DIN Nodes function as the RPs for each logical space respectively, and Steiner trees facilitate the communication among multiple DIN Nodes. Multiple powerful DIN Nodes release the burden on the single centralized server and the self-recovered DINloop avoids the single point of failure. Simulations show that DINPeer is able to reduce multicast delay for the fast service discovery.

In our future works, after the desired services are discovered, we will seamlessly integrate services running in the environment with the application running in clients.

References