A Scalable and Efficient Architecture for Service Discovery

M. Dowlatshahi  
Faculty of Information Technology  
University of Technology, Sydney  
P.O. Box 123  
Broadway NSW 2007  
Australia

G. MacLarty  
Faculty of Information Technology  
University of Technology, Sydney  
P.O. Box 123  
Broadway NSW 2007  
Australia

M. Fry  
University of Sydney  
NSW 2006  
Australia

Abstract—With the increase in the number of mobile and distributed services throughout the Internet, locating nodes capable of initiating particular services, or those able to provide particular services or resources is becoming a critical issue.

This paper presents a new architecture for service discovery and resource allocation. The architecture adopts a hierarchical tree structure for participating nodes distant from the Internet backbone, and uses a single peer-to-peer structure for service discovery at the root layer of the underlying tree structures.

The key characteristics of the proposed architecture are optimal search for both distant and close services, minimal overhead traffic, scalability, robustness, and easier QoS support.

I. INTRODUCTION

In recent years wide-area service discovery protocols have become increasingly important in the areas of Active Networks, Global Computational Grids and mobile applications.

Using a wide-area service discovery, clients requiring specific services or resource capabilities have the capability to locate where they can access a service, or alternatively where to instantiate a new service if one is not already running.

Recently two major service discovery architectures have emerged. Peer-to-peer (p2p), exemplified by systems such as Pastry [1], CAN [2], Chord [3], and Tapestry [4], and hierarchical architectures, including Globe [5], SLP [6] and Secure Service Discovery Service (SDS) [7], [8].

Both p2p and hierarchical architectures have their advantages and disadvantages. Despite their scalability and robustness for advertisement and discovery of services, p2p approaches have large traffic overheads for maintenance of the search and advertisement databases. P2p search mechanisms are also sub-optimal when a client and a service being requested are topologically close to each other. P2p structures treat all nodes equally, resulting in bottlenecks in nodes with smaller link capacity, processing power or memory size. In the p2p structure, each new participating node is inserted irrespective of the topology, and participating nodes are unaware of free processing capacity of other nodes, making resource allocation for dynamic services impossible.

Hierarchical architectures can have the advantage of using topology-based structures, and therefore provide more optimal search mechanisms for proximity located services. Hierarchical architectures can be constructed based on the topological distance from the backbone, with upper tiers receiving load distribution advertisements from the lower tiers. The main disadvantage of the hierarchical architectures is limited scalability at upper tiers of the hierarchy.

In this paper, a mixed architecture, based on hierarchical and p2p architectures is proposed, which incorporates the advantages of both p2p and hierarchical structures. In the proposed approach, a hierarchical architecture is adopted for linking nodes that are topologically close to each other but distant from the Internet backbone, and a p2p architecture is chosen for service advertisement and discovery between the root layer nodes of the hierarchical structures.

In the next section we present the related background and the design concepts of our approach. This is followed by a more detailed design. A comparison of the advantages and disadvantages of p2p, hierarchical and the proposed architectures are presented. We show effectiveness of the proposed architecture by presenting the experimental results for both p2p, and proposed architecture and conclude by describing the future work.

II. OVERVIEW

Here a brief background description of both p2p and hierarchical architectures is given.

A. P2P Structures

A p2p mechanism allows messages to locate objects (services and resources) and route to them across an arbitrarily sized network, while using a routing map at each hop. Recent examples of this are CAN, Chord, Pastry and Tapestry. Each p2p node can take on the roles of server (where objects are stored), router (which forward messages), and client (origins of requests). Objects and nodes have names independent of their location and semantic properties, in the form of random fixed-length bit-sequences with a common base (e.g., 40 Hex digits representing 160 bits). The system assumes entries are roughly evenly distributed in both node and object namespaces.

In p2p structures, each node uses local routing tables at each node, to incrementally route overlay messages to the destination ID.

Each object is associated with one or more p2p location roots through a distributed deterministic mapping function. To advertise or publish an object, the server storing the object sends a publish message toward the location root for that object. At each hop along the way, the publish message stores location information in the form of a mapping Object-ID, Server-ID. Note that these mappings are simply pointers to the server where the object is stored, and not a copy of the object itself. Gnutella [9] as a different p2p approach has based its search algorithm on flooding. Similar to other p2p approaches, Gnutella inserts nodes irrespective of their topological locations.
B. Hierarchical Structures

In the hierarchical structure, the wide-area environment is divided into domains. More specifically, the domain partition is hierarchical, i.e. one domain consisting of multiple domains at the next lower level e.g. Globe, SLP and Secure Service Discovery Service [7], [8].

Generally there are two key issues with hierarchies: (1) aggregation of service advertisements as they are propagated up a Discovery Service (DS) hierarchy, and (2) distributed service query processing (routing) in the DS hierarchy. Distributed service query processing (routing) involves the forwarding of a service query in the DS hierarchy, in order to find the Service Domains that store and provide the service.

III. DESIGN CONCEPTS

The main components of the proposed architecture are a top-level p2p structure, which connects the underlying tree structures of the service domains, Fig. 1. The nodes located in topological proximity of each other comprise the hierarchical (tree) part of the proposed architecture. Each tree structure consists of several service domains with parent-child relationships. Each Service Domain has a Service Control (SC) Server which links to service control servers of attached service domains.

In the proposed architecture, SC servers of the root domains of the tree structures comprise a single p2p discovery structure for global services.

A global service is a service that is accessible beyond a single tree. Examples of global services include IP Telephony or other protocol gateways, shared files accessible over the Internet, etc. Examples of local or private services are printers, locally shared files, etc. Each root layer SC server advertises global services from its underlying service domains to the p2p structure. A root layer SC server also forwards queries from other root domain SC servers to its underlying tree structure when receiving a query for a service in its underlying tree.

Our design attempts to meet goals of scalability, optimum service advertisement and discovery, ease of security control, fault-tolerance, QoS control and resource management.

A. Service Advertisement

In the proposed architecture, bloom filters [10] and p2p [1] mechanisms are used for advertisement of services respectively in the tree and in the p2p parts of the proposed structure. Adoption of bloom filters minimizes advertisement overload in the upper tiers of the trees. For local services, which are accessible by some or at most by all of the service domains in the same tree structure, only the changes to the bloom filters are advertised to the parent node.

For global services, in addition to the advertisement of the changes in the bloom filter vector, a description of the global service is advertised towards the root domain of the tree.

Each SC server on the path to the root of the tree, will update its bloom filter vector, and forward the complete message to its parent SC server.

Using a p2p hash function the tree root domain SC server generates a unique ID for the advertised global service. Later, the p2p routing algorithm routes the advertisement through the p2p mesh, storing the service description at each node encountered on the path to the object’s root.

B. Service Discovery

The service discovery mechanism is based on using a tree search mechanism, when both client and service are located on the same tree, or using both tree and p2p search mechanisms when client and service are on different trees.

Discovery of a service begins with obtaining the corresponding bloom filter bits for the requested service, and checking of the local and cached services by the local SC server. We assume that a client provides the parameters according to a well-known standard order, in an XML format. If the service is not available in the local domain, bloom filters of each child domain are tested. In the case of the service not being found in any of the child domains, the query will be forwarded to the parent domain. This is illustrated in Fig. 2(1).

Each parent follows a similar search process. This search process may continue up to the root domain of the tree, Fig. 2(2). When the root SC server doesn’t find the service in its own tree, it starts a p2p search process by first hashing the service description available in the query, finding the service unique p2p ID, and adding it to the service query. The modified query is then routed towards the p2p root for that service, Fig. 2 (3), which has a pointer to the service.

![Diagram of Service Discovery Architecture](image-url)
If a service is advertised by more than one tree, different mechanisms can be used to choose among them. Selection can be based on a simple round robin mechanism, the topological distance between the service and client, or any other criteria.

The first p2p node having a pointer to the service, forwards the query to the tree root node of the service, Fig. 2(4). The tree root node forwards the query down the tree until it reaches the service’s local domain, Fig. 2(5). Finally the service domain providing the service sends its reply to the client service domain, Fig. 2 (6).

In the tree structure, each service domain has an updated image of the free resources of each of its local servers and each of its child service domains. Forwarding of a query for running a dynamic service is continued until the service server that is able to fulfill the request is found.

IV. DETAILED DESIGN

A. SC Servers

The functions of a Service Control (SC) server can be divided into the following categories:

Structure Related Functionalities: These functions are related to automatic or manual insertion of a SC server into the p2p and tree structures, establishment of TCP/IP connections with neighboring SC servers, and fault recovery.

One possible approach for insertion is based on finding the RTD from a set of landmarks in the Internet backbone [11], [12]. In this approach, based on the Round Trip Delays from a set of global landmarks, either an appropriate tree for insertion of the new service domain is found, or a new tree with the new node as the root SC server is established. The details of this approach are the subject of further research.

To insert a new domain in a tree, the new domain first finds the closest parent domain. If the parent domain has less than the maximum number of children or if the joining domain is closer to it than any of its children, the parent domain will accept the new domain as its child. Otherwise the new domain should try to join to the next closest domain at the same or lower tiers.

It is worth mentioning that new domains join the same tier until it is full. This policy optimizes the number of hops in the service search.

Finally, if a new domain has to become a root domain, then it has to insert itself in the p2p structure as explained in the p2p algorithms [1]-[4].

Fault Tolerance Functionalities: Fault tolerance in the tree structures can be achieved at the Inter-domain and Intra-domain levels.

Inter-domain fault tolerance in the tree structure is achieved by advertising both backup SC server of a service domain and its parent service domain to its child domain. In case of failure of the main SC server, a child service domain may remain connected to the rest of the discovery structure through connection to the backup SC server of its parent service domain or connection to its grandparent service domain.

Intra-domain fault tolerance is based on periodic multicast of heartbeat packets, transmitted by the main SC server. In case of failure of the main SC server, a backup SC server starts multicasting itself as the main SC server. Every client and service server later detects the new SC server and makes a connection to it. The main SC server always sends a copy of service advertisement to the backup SC server, and therefore the backup SC server always has an updated image of the local and children services.

At the p2p layer fault tolerance can be achieved by assuming more than one root service domain for every tree.

Service Advertisement Functionalities: Some of the main advertisement functionalities of the SC server are: local service registrations and maintenance (service heartbeats), updating of the child service database, and generation and sending of service advertisements to the parent or p2p domains.

Services are advertised upward in the hierarchy structure. Upward advertisement reduces the overhead traffic, and results in less memory and processing overhead in the SC servers of child service domains. A separate bloom filter is allocated to each child. A bloom filter is also allocated to the local services. When advertising to the parent, the product (bitwise OR) of all bloom filters of local and child services is sent to the parent. This allows enough scalability for the upper tier SC servers.

Advertisement of global services is forwarded in the p2p structure up to the p2p root for a service. For global services we assume a standard XML format for description of the parameters. The unique ID for each global service is calculated based on a standard format of the service description.

Advertisement of each global service requires update of the bloom filter bits, and forwarding of the service description up to the root layers of the tree. Advertisement of a global service also includes the address of the originating SC server.

Computational resource advertisement contains both the sum of available resources under each service domain and the maximum capacity of the computational resource available for a single request. At the p2p level, computational resources within a tree structure are advertised with the p2p identifier of the tree root domain’s SC server.

Resource Management Functionalities: Resource Management is mainly concerned with proper server allocation for dynamic services based on load distribution and topological distance between the client and service domains. Resource management uses load distribution of the local service servers and child/ren service domains as the input. In the proposed structure, each service domain advertises overall load of itself and its children domains, when the overall change is larger than a threshold.

Service Discovery Functionalities: Different functionalities related to handling service discoveries are: reception of the queries, searching of the local, child/ren and cached services, forwarding of the queries to the parent/s, forwarding of the query in the p2p structure, caching of the found services, and replying to the requesting service domain and the client.
A. Clients

The main functionalities of a client are finding the SC server for its local service domain, connecting and registering with the SC server, and sending queries and receiving replies. Clients may only access the advertised services through their local service domain. Each client should register with its local SC server before it can send any request for a service. Clients compose new queries based on a standard format for the requested service. A client may add detailed information about the requested service to its query. It is up to the service to use this detailed information when matching.

B. Services

Services are software or software/hardware entities. The main functionalities of a service in relation to the service discovery structure are: finding the SC server for its local service domain, connecting and registering with the SC server, advertising the service and processing requests. Each new service must register with its local SC server to become accessible. A standard advertisement format allows consistent query matching against the stored service advertisements.

Each service advertises its status to its service domain both periodically and event based, to allow the SC server to have an updated record of the status of the service.

Services can be categorized as transient or persistent. Transient services perform specific actions, existing only for the duration of a single session. They are deployed as required and include such services as data compression or multimedia streaming. Persistent services however, require more specific software or hardware resources, and are more likely to require administrative maintenance or support. They can be installed manually and usually exist for long durations. Persistent services include IP Telephony Gateway or Speech Recognition. In contrast to transient services, it is expected that the number of persistent services will be relatively small.

V. COMPARISON OF ARCHITECTURES

This section presents a comparison of the proposed architecture with p2p and hierarchical architectures, with regard to several key characteristics of discovery services.

A. Search Length

One of the main factors affecting the search length in a p2p structure is the number of participating nodes (N). P2p structures are able to guarantee finding of resources in a log N number of search hops. The search length (log N) is non-optimal when the service and client are topologically close to each other.

In hierarchical tree structures, tree depth, number of services, and probability of false hits are the main factors affecting the search length. In order to limit the probability of false hits (e.g., P < 0.01), the bloom filter vector size has to be greater than 10 times the product of the number of services and the number of bloom filter hash functions [10].

In small depth tree structures with N nodes, due to a low probability of false hits, services advertised by neighboring domains can be found optimally in (log N) search hops.

In the proposed architecture the search length for finding local and global services is respectively equal to the search length in a tree and p2p structure, and therefore is optimal in both cases.

B. Scalability

In comparison to the tree structure, scalability of the p2p structure is better since it allows an almost uniform distribution of the service advertisements among all the participating nodes. In a purely hierarchical structure, with only a single root domain, the bloom filter vector size would have to be large enough to avoid false positives from occurring at the root domain.

The main advantages of the combined p2p and hierarchical structures are:

- Reduction of the number of participating nodes in the p2p structure, by several degrees of magnitude, resulting in fewer search hops at the p2p level and minimizing maintenance overhead traffic.
- Reduction of advertised services at the p2p layer by restricting advertisement of global services only, resulting in much less service update and maintenance overhead.
- Division of the Internet into many underlying tree structures with small number of nodes allowing the use of small bloom filters while keeping the probability of false hits negligible.

It should be emphasized that the tree structures in the proposed hybrid architecture are scalable because every participating node has up to a maximum number of C+1 bloom filters, where C is the maximum number of children.

C. Overhead Traffic and Processing

Overhead traffic in p2p structures is high due to periodic advertisement of services and participating nodes. This may result in large overhead utilisation at the nodes with lower link capacity, and in the worst case may result in bottlenecks for the discovery of services. In tree structures event-based advertisement of services can reduce traffic and processing overheads.

Significant overhead traffic is created in all the p2p schemes by joining and leaving several nodes and services at the same time. This may cause the p2p structure to become very slow and inefficient. In the proposed architecture the use of tree structures at the lower layers of the hierarchy only allows the advertisement of global services to the p2p layer structure. Global services are expected to be more stable, and are much fewer in number, and therefore traffic and processing overhead in the p2p layer of the proposed architecture should be significantly reduced.

VI. EXPERIMENTAL RESULTS

In this section, we present experimental results obtained with a prototype implementation of the proposed structure. We compare the obtained results with those obtained from implementation of Tree and PASTRY structures [1].

Each node is assigned a location in a plane. Coordinates in the plane are randomly assigned in the range [-10000, 10000].
Performance is compared based on the average relative traveled distance, average number of visited nodes and relative average variance of traveled distance.

In the proposed architecture, number of nodes in the p2p layer structure and number of nodes within each tree are chosen to be close to the integer part of the square root of N, total number of nodes. Within each tree, closest node to the plane center, (0,0), is chosen as the tree root node. Every root node also joins the p2p layer structure. Nodes with closest angular position in the network plane join to the same tree. Each tier within a tree is filled before a new tier is added, e.g. for a base=16, 16 nodes should join the root node as its children before next lower tier is added. This policy optimizes the number of visited nodes in the tree structure of the proposed architecture.

In the P2P structure, the entries in the routing table of each Pastry node are chosen to be closest to the Pastry node.

Also in the p2p structure, each node on the route of a service advertisement message, registers the service in its database. This advertisement policy minimizes the search length for a service.

A. Routing Performance

Routing performance is measured based on the average traveled distance and average number of visited nodes.

Traveled distance is the sum of Euclidean distances between consecutive nodes visited along the route in the network plane. Travel starts from the node receiving the query (source) and ends with the service server (Tree, Hybrid) or p2p root of the service (Pastry). Fig. 3 shows the average relative traveled distance while traveling towards the service server.

Here Relative Distance is the ratio of the traveled distance to the direct Euclidian distance between the source and service server nodes. These average relative distances are presented as a function of network size for Tree, Pastry, and the proposed hybrid architecture, (Hybrid).

Results show that the average traveled distance is minimum for the proposed hybrid architecture. In this simulation the effect of bloom filter false positive hits have not been considered.

Fig. 4 shows relative variance, the ratio of the average variance of the traveled distance to the average variance of the direct Euclidean distance between the source node and service server node, for all of the three architectures.

Relative variance of the traveled distance in the hybrid architecture is marginally greater than variance for the tree structure but smaller compared to variance in the Pastry. Smaller delay variance represents a more consistent search length from any point within the network plane.

In Hybrid 2 after finding the root node of the service tree, query travels within the service tree until it reaches the service server node. In Hybrid 1 service server instead of root node of the service tree is registered at the p2p layer. This scenario proves that for both the proposed and pure p2p architectures, number of hops from the source node to the p2p node registering the service, are equal. Tree architecture shows the worst performance in terms of number of hops.

It should be emphasized that these experiments consider only performance results for global services.

For local services search is limited to a single tree where number of nodes is close to the square root of total number of nodes, and therefore performance of the proposed architecture is far better.
It is worth mentioning that the obtained results are consistent with the results shown in [1]. The number of hops in a network with 32768 nodes is about 4.5, which is approximately equal to $1 + \log(N)$. The extra hop is attributed to the additional travel towards the service server, after reaching the p2p root node registering the service. This additional hop also explains the difference of the results shown in Fig. 3 to the average relative distance in [1]. Furthermore the difference of 0.25 hops to the exact 3.75 (Logarithm of 32768 in the base of 16) is the result of registering the service in all the nodes visited during the advertisement.

VII. FINAL REMARKS

We have presented a more efficient architecture suitable for service discovery and resource allocation in the Internet. In a Euclidean network plane, compared to p2p and tree, the proposed architecture achieves a noticeably smaller average search length. Results have also shown that search length in number of hops for both pure p2p and proposed architecture, up to the p2p node registering the service, are equal.

It is further believed that in more realistic scenarios, with non-homogeneous nodes and links, using the proposed architecture would result in even greater improvement of search delay.

In addition to a better search length, the proposed architecture has other important advantages, among which lower overhead traffic, better capability of resource management, and easier security control have been mentioned.

The main disadvantage of the proposed hybrid architecture is the complexity involved in its construction. A new participating node has to decide whether to join to the p2p layer or to one of the existing trees.

The next step in completion of this work is to design and develop efficient algorithms for construction of the proposed hybrid architecture. Further complexity and performance measures of the construction algorithms have to be investigated.

Despite existence of different algorithms for this purpose, none of them seem to be appropriate for the proposed architecture. Although attractive as a simple insertion approach, use of the binning scheme as suggested in [11] may result in a few tree structures with large number of service domains, which will not be optimum.

ACKNOWLEDGEMENT

This work has been funded by British Telecommunications as part of the ALPINE project.

REFERENCES


