A Flexible Hybrid Architecture for Management of Distributed Web Service Registries

Afiya Kassim[1], Babak Esfandiari[1], Shikharesh Majumdar[1], Laura Serghi[2]
[1]Department of Systems and Computer Engineering
Carleton University, Ottawa, CANADA
{akassim, babak, majumdar}@sce.carleton.ca
[2]Alcatel-Lucent, Ottawa, CANADA
laura.serghi@alcatel-lucent.com

Abstract

This paper proposes an adaptive architecture for Web service registries that changes topology based on system state and user requirements. Three distributed registry topologies are presented and analysed based on their delay and network traffic. An adaptive registry deployment architecture is proposed that uses a communication overlay layer. The interconnection topology is changed based on the state of the system.

1. Introduction

In order for web services to be shared over the Internet, the service providers publish the required components in a service registry whose location is known to service requesters.

Distributed registries provide advantages such as scalability, no single point of failure, as well as no performance bottleneck on one centralized component. Such a system is also required when distributed registries already exist in a federation and we have to integrate these registries. This research concerns Extranets that connect the users in a company and its partner companies and can span multiple gateways. Each company in the Extranet has its own UDDI registry, thus giving rise to a distributed registry system for the entire Extranet. A distributed registry deployment architecture is required in this case.

A meta-directory-based design is proposed. The meta-directory node stores the location of a Web service registry that is capable of responding to the user query. The communication between the meta-directories adapts to user requirements and system state by having a communication overlay layer on every meta-directory node.

This paper presents the high level architecture for the adaptive distributed service registry. Work on implementing the prototypes for this architecture and on evaluating its performance is currently underway.

2. Background

Deployment architectures can be divided into three main categories: centralized, decentralized and hybrid architectures.

The centralized architecture involves the existence of a single centralized registry, which stores all the registry entries.

The decentralized architecture often involves the existence of peer registries that introduce autonomous behavior. The peers have equal responsibilities and they act as both service providers and requestors [2]. The architecture is dynamic as peers can join and leave the network without disrupting service.

The hybrid architecture incorporates features of both the centralized and the decentralized architectures. This is done by introducing super peers or root peers that provide a form of infrastructure within the nodes while peer-to-peer operations are maintained within the clusters [1]. The super peers provide transparent registry access while the web service provider and requester are not aware of the distributed nature of the registry.

Even though decentralized approaches provide the most scalable solution, there is a trade off with communication overhead as requests are forwarded to multiple nodes in the peer network. The hybrid architectures provide a better solution in terms of scalability as they reduce the communication overhead but in turn they introduce an additional administrative overhead.

While the briefly introduced architectures do take into account scalability and latency issues, none of them are optimal for every network as can be seen in the analysis in Section 3. We propose a system that offers the human operator the possibility of selecting

Fifth Annual Conference on Communication Networks and Services Research(CNSR'07)
0-7695-2835-X/07 $20.00 © 2007 IEEE

the most appropriate architecture given the particular system state.

3. Analysis of deployment architectures

This section compares CHORD, fully connected, and Super Peer topologies in terms of performance. A CHORD [3] network uses a hash function that enhances the lookup protocol by providing flexibility in keyword searches using a Distributed Hash Table (DHT). In a DHT, the keys are partitioned among participating nodes and messages can be efficiently routed to the unique owner of any given key.

A Fully connected network is the one in which each node is directly connected to all the other nodes and requests are broadcasted in the system.

The Super Peer network used in this research has clusters with each cluster having a super peer and a maximum of 5 nodes per cluster. CHORD is used within the clusters, while the cluster heads are fully connected.

Fig. 1 compares the results for the total number of hops during a search, which determines the delay. The total number of hops needed to find a result to a query in a fully connected network is always equal to 1. For a CHORD network the total number of hops is equal to $\log_2 N$ [3] where N is the total number of nodes. For a Super Peer network, when a request arrives in a cluster, the query is first forwarded within the cluster using CHORD. If the registry nodes within the cluster do not have the service requested, the cluster head broadcasts the request to the remaining cluster heads. A search is then performed within the other clusters concurrently and the response is returned to the requesting cluster head.

The total number of messages exchanged during a search, that determines network traffic, is compared for the different topologies in Fig. 2. The number of messages sent in a fully connected network is always equal to the number of nodes in the network. A CHORD network on the other hand sends a total of $\log_2 N$ messages [3]. According to the flow of messages as discussed in the previous paragraph for a super peer network, the total number of messages exchanged is equal to the sum of the messages exchanged in the CHORD clusters plus the number of messages in the broadcast between the cluster heads.

CHORD maintains stable routing tables by exchanging periodic messages that update the routing information. These messages introduce network traffic as indicated in Fig. 3.

According to the figures, it can be seen that the performance of all the architectures varies when the number of nodes in the system changes. For example, a fully connected architecture performs better in smaller networks as it provides minimum latency as well as minimum administrative costs. But when the number of nodes increases, a fully connected network introduces high network traffic.

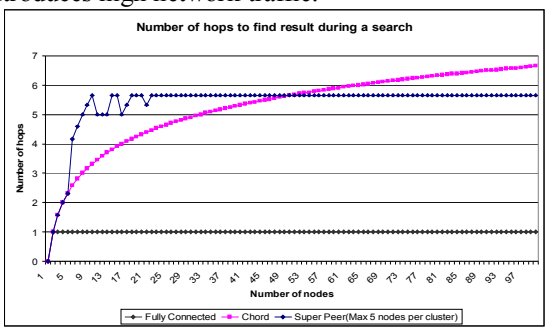

Figure 1. Total number of hops for different network topologies

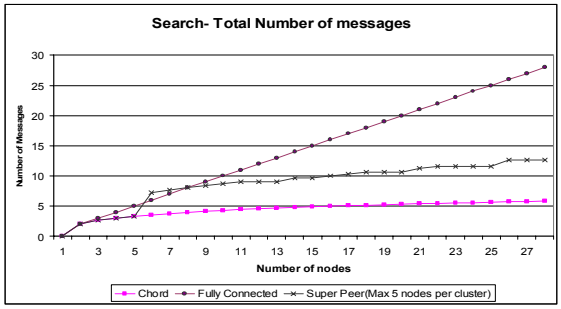

Figure 2. Total number of messages for different network topologies

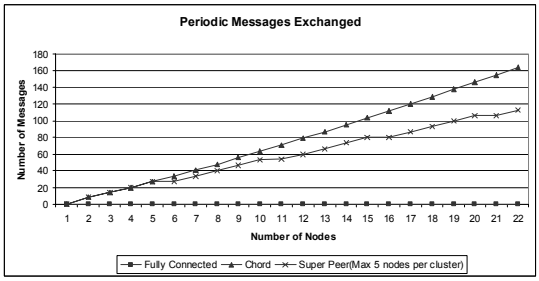

Figure 3. Periodic messages exchanged for different network topologies

It is preferable that a user can decide which architecture to adapt to based on the system state. A user can manually change the topology based on system state or use a set of two tuples $<N_i, T_i>$, containing a threshold value (N_i) and a topology (T_i), that the system can use to automatically adapt to the appropriate topology. For example, as soon as the number of nodes reaches N_i, topology T_i is used by the communication overlay layer.

To the best of our knowledge there is no existing registry architecture that adapts its topology based on the number of nodes in order to improve system performance.

The next section presents architectures that can support multiple network topologies.

4. System design

4.1. Meta-directory-based design

In a meta-directory based system, the meta-directory stores the location of the registry that contains the desired Web Service. A search in the meta-directory is performed first. The user then queries the registry containing the Web Service.

4.1.1. Structure. The meta-directory-based design separates the UDDI registries from the searching algorithm. Fig. 4 gives an illustration of the interactions between the users, the Extranets and the meta-directories. A communication overlay is used at a meta-directory node that is connected with other meta-directory nodes in the system. The communication architecture is selected from fully connected, CHORD, and Super Peer based on network parameters (such as number of nodes and delay).

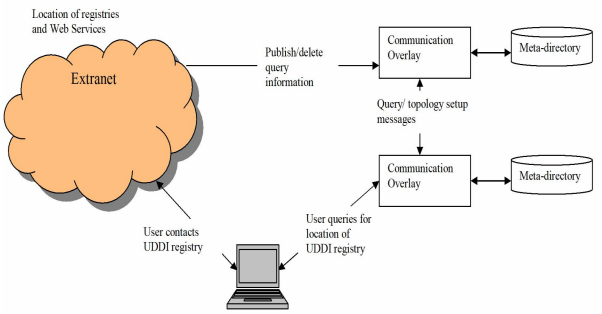

Figure 4. Meta-directory structure

4.1.2. Hash table structure. The meta-directories store a query parameter as the hash table key and the key value is the location (IP address) of the UDDI registry that can answer the query.

Each meta-directory is responsible for a range of hash values and is given a unique identifier and arranged in a ring (see Fig. 5). Each hashed key is assigned to the first peer (successor) following it on the ring in a clockwise direction, as in key 5 is assigned to the meta-directory with ID 8.

If a company called "Software Solutions" publishes a web service called "Order Product" in one of its registries, under the category "Order", the registry forwards the published information to one of the meta-directories.

The meta-directory receiving the publish request parses the message and the attributes are then hashed. If the business name, service name and service category values hash to 3, 5, and 10 respectively, the values will be forwarded to their respective successor peers as detailed in Fig. 5.

Using hash tables as in the CHORD algorithm ensures that the meta-directory node receiving a query knows which meta-directory node has stored that hash value which minimizes latency.

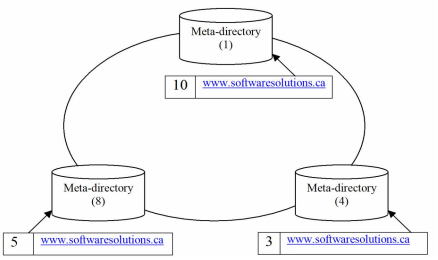

Figure 5. Hash values distribution

4.2. Alternate directory design

We have also started investigating an alternate design whereby instead of using meta-directory nodes, the registry nodes directly communicate with one another. Unlike the meta-directory-based design, querying is not abstracted from the registries.

A distinct advantage of the alternate design is that it does not require additional nodes for housing the meta-directories. A comparison of the two architectures both in terms of their applicability to real Extranets and performance forms an important direction for future research.

5. Conclusions

This paper investigates distributed Web Service registry topologies and analyzes the performance of three distinct distributed topologies. A meta-directory-based architecture is proposed. An alternate design that does not require additional meta-directory nodes is also under current investigation. The next step in the dynamic configuration registry architecture is the implementation and testing of the proposed architecture.

References

[1] Z. Du, J. Huai, Y. Liu, "Ad-UDDI: An Active and Distributed Service Registry," in *Proceedings of the 31st International Conference on Very Large Databases (VLDB'05)*, Workshop on Technologies for E-Services, Trondheim, Norway, August 2005.

[2] C. Schmidt and M. Parashar, "A Peer-to-Peer Approach to Web Service Discovery," World Wide Web Journal, Vol. 7, Issue 2, June 2004, pp. 211-229.

[3] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup services for internet applications," in *Proceedings of the Special Interest Group on Data Communication (ACM SIGCOMM'01)*, San Diego, CA, August 2001, pp. 149-160.