Enabling Peer-to-Peer SDP * in an Agent Environment

Naveen Srinivasan
University of Maryland Baltimore County
Baltimore MD 21250
nsrini1@cs.umbc.edu

Timothy Finin
University of Maryland Baltimore County
Baltimore MD 21250
finin@cs.umbc.edu

ABSTRACT

Negotiation, co-operation and collaboration between multi-agents systems is the key to success in the agent world. An agent that is capable of these actions can offer them as services. The main challenge lies in discovering the agents that provides such services. FIPA architecture facilitates service discovery process with the help of Directory Facilitator (DF), but the service discovery process is limited to a particular Agent Platform. Federation between the DFs would lead to a Gnutella type peer-to-peer network, which is neither scalable nor efficient. In this paper we propose modifications to existing Directory Facilitator to enhance the global service discovery among the agent platforms. We propose a scalable architecture, which forms decentralized clusters of Directory Facilitators. These clusters are self-governing, fault-tolerant and support global service discovery. We also discuss ways of improving the performance of the service discovery process using this architecture.

Keywords

Agent Platform, Directory Facilitator, peer to peer network, clusters, leader election

1. INTRODUCTION

The capabilities of negotiating, co-operating and collaborating with other agents in an open system increases the overall efficiency of the system. The challenge is to discover the agents that have these capabilities and offer them as services. Directory Facilitator [2] (DF) of the FIPA agent platform [2] is a repository where agents can register the services that they provide. Agents can also search the repository for desired services but the search is limited to that particular agent platform.

With the increase in deployment of agent platforms, services provided by agents will increase at a faster rate. Since communication between the FIPA agent platforms is possible, the service discovery should not be restricted to a particular agent platform. Agentcities project [5] has defined an infrastructure where agent platforms can register themselves and participate in forming a global network of agent platforms. Since the platform address can be obtained using this infrastructure, agents can directly contact the DFs in other agent platforms and search for required services. However, this approach limited by the number of addresses that can be queried at a time.

The Distributed Model approach exploits FIPA's notion of DF federation. In this approach, each DF forwards the service discovery queries to DFs that the user has requested to federate with. We can enhance the performance of this approach by eliminating duplicate queries and limiting query flooding in the network. After these improvements the system looks similar to the Gnutella [3] peer-to-peer network, which is not scalable [10]. We will discuss more about Gnutella in related work.

A central repository like Agentcities.net [5], maintains the services provided by the agents in a network of agent platforms and acts like a global Directory Facilitator. In a dynamic environment[1] the central repository will have stale information unless the agents update their information at the central repository. This increases the burden on the central server that has to process these updates in addition to processing service discovery queries from agents in the environment. Although this architecture will have a fast query response time, it will require high maintenance to host the central repository.

We envision that, the DF, apart from providing yellow pages service to its agent platform, should also be responsible for providing global search results by federating with other DFs. We also propose modifications to the existing DF structure that will support this infrastructure. The idea behind the protocol is to take advantage of both centralized and distributed approaches.

We discuss the related work in section two. In section three we propose an architecture that provides infrastructure for global service discovery. In section four we discuss some ways by which we can exploit the cluster-based architecture followed by implementation and conclusion.

*Service Discovery Protocol

[1] agents or agent platforms enter and leave environment randomly.

2. RELATED WORK

Sycara et al. [9] proposed a solution to service discovery by deploying network of agents similar to Gnutella network. In Gnutella protocol, the query is transmitted to its "one hop" neighbors which is in turn forwarded to their one hop neighbors. The search results are also routed back along the same route as the query propagated preserving anonymity in the network. We believe that anonymity is not as important an issue in the agent world, hence search results can be directly sent to the agent which generated the query. This decreases the network traffic and improves the performance of the network. In this paper each agent is mapped to a node in Gnutella network. If we consider each agent platform as a node it will reduce the number of nodes in the network, and hence the traffic.

Sycara et al. [8] proposed the idea of middle-agents, which are the agents that help others to locate required-service providing agents . Middle-agents can also perform sophisticated actions, such as service composition. This approach is similar to the centralized model, and therefore inherits the drawbacks of the centralized model.

Another area which has drawn attention in the research is Peer-to-Peer networks (P2P) service discovery protocol. Napster [4] and Gnutella are two prominent models of P2P network. The Napster model is similar to the central repository model, whereas the Gnutella model implements a distributed mechanism. A servent [2] on entering the network announces its presence to its neighboring servents, these neighboring servents in turn forwards the announcement to their neighboring servents and so on. In response to this announcement each servent sends brief information about itself to the new servent. Similarly, for searching the resources, the query is propagated through the network and only the servents with appropriate resources respond to the query. Recent analysis shows that the Gnutella network will not scale up to the size of 1 million users [10]. Also the performance of network is brought down by free riders [3].

3. CLUSTER-BASED ARCHITECTURE

We believe that an existing DF should be modified to support infrastructure for global search. Below we address the problems in the existing DF model and possible solutions to tackle them.

Problem A : The environment we are dealing with is very dynamic. So, a DF should try to return accurate search results. In the centralized model, this can be achieved by updating the central server, but this process will result in excessive traffic near the central node. in the distributed model similar to the Gnutella network, the DFs do not require forwarding update messages . Also, for each search query, nodes perform a real time search and hence do not return any stale search results. Thus a distributed model fits well in a dynamic environment so the DFs should also be modified to form a distributed network with other DFs in other agent platforms.

Problem B : The service registration and deregistration process is under the control of agents and not the DF. We cannot depend on agents to deregister their services when they leave the agent platform. This will leave stale information in the DF. This problem can be solved by adding lease time with the service registration process. Lease time is the specific time period for which the service registration in DF is valid after which DF deregisters the service. The agent has to re-register if it is still providing that service. Though this process has an overhead for agents in the platform, the DF will not have stale service registration.

Problem C : Malicious agents can register fake services in the DF. This problem can be handled by using a reputation mechanism [11]. The DF should be responsible for hosting the reputation mechanism, so that, it can return the reputation of each agent in the search result. An agent, after receiving search result, can start interacting with the agents in the search result based on their reputations. After the interaction, the agent can send feedbacks to the DF about the agents that provided the service. The DF uses these feedbacks to modify the reputation of the agent that provided the service.

Problem D : DAML S [6], an initiative by semantic web community, provides a core set of markup language constructs for describing the properties and capabilities of Web services. It also provides facility to automate Web service discovery, execution, interoperation, composition and execution monitoring. DAML-S is built on top of DAML+OIL [1]. We believe that incorporating DAML-S ontology will enhance the searching capability of DFs. DFs need not perform complex operation like Matchmaker[8] but perform some basic inferencing with DAML-S ontology. For example, if there is a composite service registered with the DF, then it should also register the services that forms composite service individually, if they are not already registered.

We propose a cluster based architecture that is distributed, self-governing and has faster query response time. We assume that DFs of each agent platform knows few other DFs, and form a weakly connected network. The broad idea of this architecture is forming clusters in the network and each cluster selects one node (cluster head) to act as a proxy for the entire cluster. This proxy knows few other proxies in the network and forms a distributed network like Gnutella, while a centralized model, like Napster is formed between the cluster nodes and its proxy. Figure 1 shows the architecture. Though proxies form a Gnutella type network, the number of nodes involved are comparatively less when compared to number of nodes in the network, and hence the traffic generated by the network will be less. Since the proxy responds on behalf of the cluster the query response time is faster. An overhead of this architecture is that these nodes in the clusters have to frequently poll the cluster head and elect a new cluster head if they find that cluster head left the network, otherwise the cluster will be isolated from the entire network.

3.1 Assumptions

We assume some entity like Agentcities will act as a central server to which DF of an agent platform can register itself and also receive other agent platform addresses in the network (similar to central server in the Gnutella network).

[2] p2p term for a node that is both SERVer and cliENT

[3] users that dont share any resources

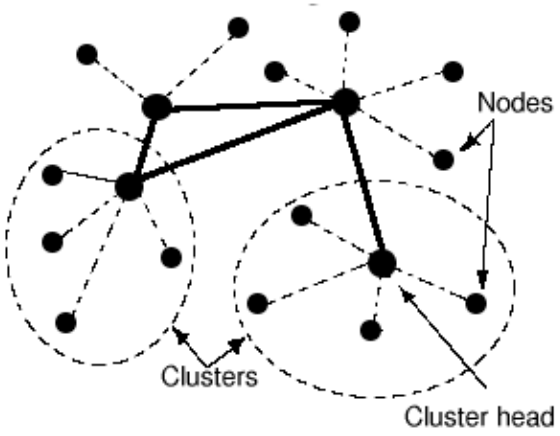

Figure 1: Cluster-based architecture

However, after the registration, the central server does not play any role in the service discovery.

3.2 The Protocol

All the DF nodes in the clusters are equal peers and any DF is qualified to be a cluster head. All the DFs create a profile which contains information, like name of DF, address of the agent platform, its preference to become the cluster head. The modified DF composes of four components. The functionality of each component in DF is mentioned below.

3.2.1 Startup

When a new agent platform comes up, the DF registers itself to the central server, by submitting its profile. As a part of registration it also requests the central server for other DF profiles. If the central server does not return any DF profile then the new DF elects itself as cluster head. If the central server returns profiles of existing DFs then the new DF contacts the existing DFs and gets profiles of their cluster heads. After receiving the cluster head profiles, the DF starts contacting these cluster heads, to get additional information like availability and preferences. Based on the information returned by the cluster head, a new DF can select the cluster, and join it. If the new DF decides to join a particular cluster then it sends its DF information (registered services) to the cluster-head and receives profiles of other DFs in the cluster from the cluster head. The cluster head also sends the profile of the new DF to the cluster nodes. In this way the cluster head has a repository of DF information of the whole cluster and the cluster nodes are aware of other nodes in the cluster.

If none of the cluster heads matches the DFs requirement, then it can either get a new set of nodes from the central server and starts the registration process again or can become its own cluster head. After becoming the cluster head, it announces its presence to other cluster heads, which it gathered in the registration process, and participate in the service discovery process.

3.2.2 Query Routing

In this section we describe how a search query is propagated in the network. When a DF in an agent platform receives a query apart from searching its own information, it forwards the query to the cluster head. The cluster head searches the DF information of the whole cluster and also forwards the query to other cluster heads it knows. The other cluster heads repeat the same process. The results are routed back to the cluster head which initiated the query forwarding, and the cluster head sends these result back to the DF which in turn forwards the results to the agent which generated the query. The results got back from the search could be cached by every cluster head for answering future queries.

Whenever a new query is received by the cluster head it tags a unique ID to the query, which is used to stop processing and forwarding of duplicate queries. The query flooding is controlled by adding a time-to-live (TTL) field. Every cluster on receiving a query will decrement the TTL field before processing the query. The agent which initiated the query can request its DF to set the TTL field high or low depending on its need.

3.2.3 Cluster Maintenance

Whenever DF's information is modified due to addition, modification or removal of service from a DF the information is updated in cluster head. The cluster head is responsible to find out if any DF (agent platform) left the cluster without notice. The cluster has to update this information to the cluster, otherwise this stale information could affect the leader election process.

3.2.4 Leader Election

Since the cluster head is the only interface to the whole cluster, we should assure that a cluster head is always up. Otherwise the cluster is isolated from the network. If the nodes in the cluster find out that the cluster head has left the network, then negotiation between cluster nodes start, and a new cluster head is elected. The newly elected cluster head announces that it has replaced the previous cluster head to its neighboring cluster head .

Next, we describe a simple leader election algorithm. Since DFs in the cluster have profiles of other DFs in the cluster, the DF which expressed the highest preference in its profile to become a cluster head can be elected. If two or more DFs express the same preference level, then we can use a simple resolution protocol, based on the platform uptime or agent platform address, to break the tie. Since leader election is expensive we should make sure to elect cluster heads which are very stable.

3.3 Exploiting the Cluster-based Architecture

In this section we discuss ways by which we can exploit the cluster based architecture and thereby improving its performance. Since the architecture is cluster-based it is very flexible in terms of moving nodes between clusters, merging the clusters and splitting the clusters.

3.3.1 Hierarchy

The architecture we discussed is a one-level architecture. We could also form a hierarchy of clusters. The top level cluster head can act as query routers instead of holding the

DF information of the lower levels. Assuming we have a two-level cluster head architecture the bottom level cluster heads would store the DF information of its cluster. Similar clusters are formed by bottom level cluster heads and a proxy is elected (these proxies form top level nodes) to represent them. The top level node forms distributed network model like Gnutella.

Now when a bottom level cluster head receives a service discovery query, the bottom level cluster head will forward the query to its cluster head in the top level, instead of flooding the bottom level cluster head network. Now the top level cluster head will forward the query to each of the node in the bottom level cluster and also propagates the query in the top level nodes. Note that the number of nodes participating in the Gnutella type network is reduced, thereby reducing the generated traffic substantially.

3.3.2 Cluster Formation

We believe that grouping nodes into clusters, based on the DF information they have or their network location will improve the performance. For example, if we group the cluster based on the location of agents, it would localize the traffic generated by cluster maintenance component.

We believe that a cluster should fall into one of following ideal categories for best performance. In the first category there is minimal update in the cluster then the cluster head will always have the correct information and will not be swamped by update messages. In the second category, the DF information of all nodes in the cluster are frequently modified, and the cluster head will be flooded with update messages. In this case, the cluster head instead of maintaining the DF collection of the cluster, can forward the queries to nodes in the cluster itself. This would reduce load on the cluster head and also result in accurate search results. However if the clusters do not fall in both these categories then some DFs can exchanged between clusters to form an ideal cluster.

We can improve the performance of the cluster by merging or splitting the clusters depending on the load on the cluster head. If many nodes leave the cluster, then we can merge two or more clusters so that fewer cluster heads participate in the service discovery protocol and hence decrease the traffic. If a cluster head is swamped due to large number of queries or due the load in its agent platform then we can either split the cluster or elect a new cluster head to replace the cluster head.

3.3.3 Caching

If we are caching the search results, then only the cluster head that generated the query will have the global search results. All the other cluster heads will have the search result of their DF collection only. If the same query is generated within the same cluster, then the cluster head can return cached results. However if the same query received from its neighbor, then there is no use of the cached global results because even though it can return the cached global result it can not stop the query from flooding the network. So, only a cluster can be benefited by caching global results. We can cache search results in multi-level architecture by forwarding the global search results to the top level. Now a cluster of clusters could be benefited.

4. IMPLEMENTATION

We have implemented the above mentioned architecture using JADE agent development toolkit [7]. We have modified the DF of the JADE agent platform to federate with other DFs as and when the platform is started. We have also added lease time functionality and semantic search mechanism to the DF. Our next effort is to produce results to prove the scalability of the architecture. We are also developing a sophisticated leader election algorithm which reflects the dynamic network load.

5. CONCLUSION

The success of a open agent system depends on discovery of the required service in the network. In this paper we proposed scalable cluster based architecture for agent service discovery in a large agent network . We proposed modifications to the existing DF to support global service discovery. Then we discussed how we can exploit the current architecture which would improve the performance of service discovery. We believe that this type of distribute approach would help in building open agent systems.

6. REFERENCES

[1] Daml+oil http://www.daml.org/2000/12/daml+oil-index.

[2] Fipa agent management specification web site. http://www.fipa.org.

[3] The gnutella protocol specification v0.4 www.clip2.com/gnutellaprotocol04.pdf.

[4] Napster protocol specification http://opennap.sourceforge.net.

[5] The agentcities project web site. http://www.agentcities.org. 2001.

[6] S. et al. Daml-s: Semantic markup for web services. *International Semantic Web Working Symposium (SWWS)*, 2001.

[7] G. R. Fabio Bellifemine, Agostino Poggi. Jade : A fipa-compliant agent framework. *Proceedings of Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 97–108, 1999.

[8] K. S. Hao-Chi Wong. A taxonomy of middle-agents for the internet. *Fourth International Conference on Multi-Agent Systems*, 2000.

[9] P. M. S. K. Langley, B. Discovery of infrastructure in multi-agent systems. *Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, 2001.

[10] A. I. Matei Ripeanu, Ian Foster. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system.

[11] D. M. Roger Dingledine, Michael J. Freedman. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, Nov 2000.