Adaptive, Distributed Resource Discovery for ALAN and Grid Computing

Ioannis Liabotis
Departement of Electrical and Electronic Engineering, UCL
{i.liabotis@ee.ucl.ac.uk}

Abstract

In a heterogeneous system such as a wide-area Application Level Active Netowrk (ALAN) or a computational Grid, different types of resources exist and resource availability is not implied in every node. A basic service in this type of environment is the resource discovery service. The discovery service is responsible for finding resources and forwarding the user request to the nodes that can handle it. This paper presents some ideas for a totally distributed resource discovery protocol for application level active networks (ALAN) and Grid Computing environments. The protocol is based in the ideas of peer-to-peer computing. All the nodes that implement the protocol act autonomously based on information gathered by a limited number of neighbor nodes. This autonomous node behavior is envisaged to provide self-organization to a wide-area system of a local area network.

1 Introduction

Application level active services are based on programs supplied by the users of the network. Those programs will run on equipment owned by the operators, enabling users to have access to custom services that will be managed by them without the operators intervention. Furthermore, computational Grids provide mechanisms for sharing and accessing large and heterogeneous collections of geographically distributed resources. In a heterogeneous distributed system the different types of resources exist and resource availability is not implied in every node.

A resource can be a hardware resource such as CPU cycles, memory/storage or network resources. The availability of such resources depends on the hardware specifications and the resource requirements of applications that are already running on those resources. Furthermore, a resource can be a specific management interface or software services running on a particular node.

One property of all the three types of resources, is the fact that they appear and disappear dynamically in the grid network.

A basic service of an ALAN enabled network or a Grid environment is the resource discovery service. Users request a number of resources with specific properties in order to run their applications. The discovery service is responsible for finding those resources and forwarding the user request to the nodes that can handle it. In an environment where thousands of nodes and hundreds of users are going to use the resource discovery service a centralized mechanism would possibly fail [BAG98]. Both in the cases of a wide-area grid and that of a computer cluster a decentralized protocol can be used to provide self organization.

A peer-to-peer protocol for resource discovery in a grid computing environment is proposed. It is envisaged that small world like connectivity between peers would improve the efficiency of the discovery protocol.

2 Related Work

The resource discovery problem has been addressed in various different application contexts. In the context of Grid computing several projects have addressed the resource discovery and resource scheduling problem using centralized, hierarchical or distributed algorithms. Condor [LLM88] uses a centralized *collector* that provides a resource information store, that listens for advertisements of resource availability. Globus [FK97], is using local resource managers (GRAM) to update a directory service with information about the availability and capabilities of managed resources. The directory service (MDS) is based on LDAP using a hierarchical, tree-structured name space. The *Data Grid* project [HJS+00] uses a decentralized, query based resource discovery mechanism and a hierarchical organized scheduler based on LDAP.

A. Iamnitchi and I. Foster in [IF01] propose a fully decentralized resource discovery mechanism for Grid environments. The mechanism is based in a peer-to-peer architecture using request forwarding algorithms. They

evaluate four different request forwarding mechanisms: Random forwarding, experienced based + random, best-neighbor (based on the node that replied the maximum number of requests, and experience based + best-neighbor. They evaluate the different forwarding policies using a simulation model trying to identify the trade-offs between communication costs and performance of the system.

3 Using Peer to Peer Protocols for Resource Discovery

The proposed solution for resource discovery, is a fully distributed protocol, which is based on the ideas of peer-to-peer computing [O'R]. The protocol does not guarantee to find the best available resources but it tries to identify an acceptable solution that will be close to the optimum. Each node of the network acts autonomously in order to identify the best possible node that can serve a specific request.

Each node is connected to a number of other nodes called *neighbors*. Initially the neighbors are the nearest topologically neighbors (based on geographical location and bandwidth availability between neighbor nodes), and a few "far" neighbors. The use of a two level control mechanisms, is proposed. The first one is *query-based*, while the second one is *advertisement-based*.

The *query-based* mechanism will be invoked when an incoming request cannot be *resolved* by the node that the request is received. When the recipient node does not have any information about the resources needed by the request or the information is relatively "old", a query is sent to the neighbor nodes. Associated with each query is a *query time to leave* (*QTTL*), that determines the number of times a query can be forwarded from a node to its neighbors. The QTTL will limit the traffic generated by the protocol. The trade-off between the generated traffic and the success rate of the protocol has to be studied.

When there is information about the node that can handle a specific request the node simply forwards the request to the appropriate node or executes the requested application if the resources are available locally. Associated with each request is a *forwarding time to live* (*FTTL*), that determines the number of times a request can be forwarded until it has to be serviced. The FTTL is used similar to QTTL for controlling the generated traffic and the success rate of the protocol.

The second control mechanism is *advertisement-based*. It is activated when a new node appears and advertises its resources. It is also initiated when a dynamic resource, such as CPU availability is changing state. E.g. when a node becomes underutilized it can send an advertisement to all of its neighbors with information on the resource availability. On the other hand if a resource becomes unavailable a node can send a *withdrawal advertisement* informing its neighbors about the lack of local resources. Advertisements have an *advertisement time to leave* (*ATTL*), that determines the number of hops that the advertisement has to be forwarded.

4 Information Distribution and Caching

Information about the resource availability in the network of nodes is distributed using the *query-replies* and *advertisements*. This information is cashed in the nodes that receive the replies or the advertisements. Initially when no past information is available queries are forwarded to the initial list of neighbors. When a reply is received the id of the node that can handle the request is cached in a list of nodes that provide the requested resource, together with more specific information on those resources. The list of available resources is also updated each time an advertisement is received. Nodes that are not included in the future replies messages, are removed from the list after a specified time period or when the list exceeds a specified size.

Using a different cache for each type of resource, different *virtual* network topologies are created for each different resource. Figure 1 illustrates the generation of virtual topologies. Initially the node are connected to its nearest neighbors (topologically) and some random far neighbors creating a small world topology. More information about small world topologies is provided in Section 5. The list of neighbors is changing giving separate lists of neighbors for the different types or resources, thus generating the different virtual topologies. Members of the virtual topology are nodes that have the specific resources that the virtual topology identifies or "know" about the requested resources.

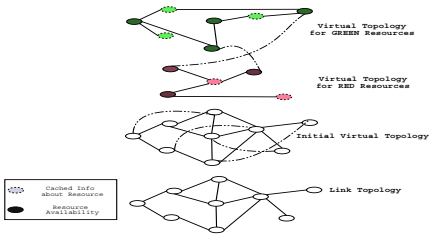


Figure 1: Virtual Topologies for different types of resources

When there is cached information available, requests and queries are forwarded to nodes that belong to a specific virtual topology. In order to avoid clustering between virtual topologies, nodes keep a number of connections to nodes that belong to different virtual topologies. Thus queries and advertisements are sent to a number of nodes that can be found in caches of other resource types. Another mechanism that resolves the clustering problem is the advertising mechanism that the nodes can use.

5 Small World Connectivity

Small world connectivity [Wat] has been observed in various types of networks. For example social networks, the world wide web, and neural networks are all examples of networks that exhibit the *small-world* topology. The main characteristic of small-world networks are the small average length and the large clustering coefficient. The small average length is beneficial to the resource distribution protocols because with a small number of forwarding hops a big portion of the network can be covered. The clustering coefficient captures how many of a node's neighbors are connected to each other. A large clustering coefficient groups together nodes with specific characteristics.

For the purposes of resource discovery, small world connectivity is maintained in two levels. Firstly in the topological level, a node is connected to a number of "near", based on the link topology, nodes. Furthermore a node is connected to a few "far" neighbors. On the second level the small world connectivity is maintained between different virtual topologies. Each node is linked with many nodes in the same virtual topology, but maintains a small number of connections to nodes that belong to different virtual topologies.

6 Simulation and Preliminary Results

A limited version of the resource discovery protocol has been simulated using a custom simulator developed in C++. The simulation assumes one type of resource (CPU), and the requests are CPU intensive processes trying to find the least loaded node. The simulation model is shown in Figure 2.

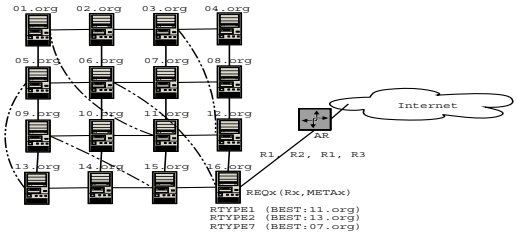


Figure 2: The simulation model

The resource discovery protocol implements the request forwarding and advertisement messages. All requests are initially sent to node 01.org (top left hand corner in Figures 2 and 4). During the simulation the mean server load and the local dispersion where calculated. Furthermore the simulator is discovering the best server with each new request arrival, using a centralized methodology that was getting each node's load. This was done in order to calculate the success rate of the distributed load distribution algorithm. The success rate is defined as the number of times a request was forwarded to the least loaded server over the total number of requests. The simulation was ran initially, with the nodes connected to their nearest neighbors only (far neighbors "*off*"), and then with each node connected to one far neighbor (far neighbors "*on*"). Figure 3 shows the relationship between success rate and the total network load, for both the cases of far neighbors "*off*" and " *on*".

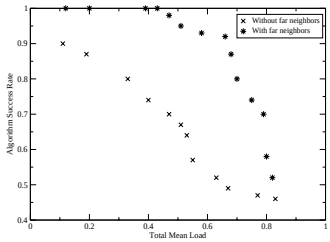
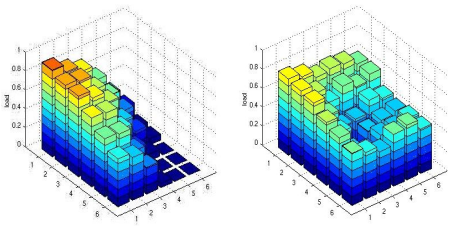


Figure 3: Algorithm Success Rate over different Network Load

Figure 3 demonstrates that increasing the network load the success rate decreases. Adding a far neighbor to each of the nodes of the network the success rates is higher even for higher server loads.

Figure 4 shows the effect of the small world connectivity to the local dispersion. Where there is no small world connectivity the load is not evenly distributed. Furthermore there is a number of nodes that do not receive any requests even though others are highly loaded. This leads to the conclusion that the network can become clustered. Clustering could be avoided by increasing the FTTL or ATTL. In the case that the small world connectivity is "on", and keeping the same *FTTL* the load is more evenly distributed giving a more uniform load dispersion.



(a) (b)

Figure 4: Local dispersion without (a) and with(b) the small world connectivity

7 Conclusions and Future Work

This paper presents a proposal for a totally distributed resource discovery protocol for application level active networks (ALAN) and Grid Computing environments. The protocol is based in the ideas of peer-to-peer computing. All the nodes that implement the protocol act autonomously based on information gathered by a limited number of neighbor nodes. This autonomous node behavior is envisaged to provide self-organization to a wide-area system of a local area network.

Important factors for the performance of the resource discovery protocol and that have to be studied are: The traffic generated by the messaging mechanisms.This traffic comes as the network bandwidth overhead that the protocol creates. This amount of traffic has to be measured and be minimized in order to have an efficient and scalable algorithm. The current simulation experiments assume that the network topology is a simple lattice network Additional simulations will be done with different network topologies that resemble the real Internet, thus testing the scalability of the algorithm.

Furthermore the best policies for the size of caches and thus the number of links that each node preserves, the time that information in the caches can be considered valid, the frequency of advertisements, QTTL, FTTL, ATTL and other parameters have to be studied.

References

[BAG98] R. Buyya, D. Abramson, and J. Giddy. Economy driven resource management architecture for computational power grids. In *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2000)*, Las Vegas, USA, 1998.

[FK97] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

[HJS+00] Wolfgang Hoschek, Francisco Javier Janez, Asad Samar, Heinz Stockinger, and Kurt Stockinger. Data management in an international data grid project. In *GRID*, pages 77–90, 2000.

[IF01] Adriana Iamnitchi and Ian Foster. On fully decentralized resource discovery in grid environments. In *International Workshop on Grid Computing*, Denver, Colorado, 2001. IEEE.

[LLM88] M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *8th International Conference of Distributed Computing Systems (ICDCS 1988)*, San Jose, CA,USA, 1988. IEEE CS press.

[O'R] O'Reilly. Openp2p.com, http://www.openp2p.com.

[Wat] Duncan J. Watts. *Small Worlds: The Dynamics of Networks between Order and Randomness*. Princeton University Press, New Jersey, USA.