Proceedings of the Seventh International Conference on Machine Learning and Cybernetics, Kunming, 12-15 July 2008

OPTIMISATION STRATEGIES FOR DISTRIBUTED COMPUTING USING AN ADAPTIVE RANDOMISED STRUCTURED NETWORK

CHUN-CHE FUNG, JIA-BIN LI

School of Information Technology, Murdoch University, Murdoch, Western Australia
E-MAIL: l.fung@murdoch.edu.au, eric_li@ieee.org

Abstract:

One way to improve computational efficiency for complex engineering applications is to utilise distributed computing. In such distributed system, accessing objects through location-independent names can improve the system's transparency, scalability and reliability. Names however need to be resolved prior to passing the messages between the objects. This paper reports an Adaptive RandoMised Structured search network termed ARMS, which utilises a distributed Ant Colony Optimisation algorithms (ACO) to improve the efficiency of searching in a distributed environment. The paper further investigates different kinds of optimisation strategies in order to improve search efficiency. Simulation studies have shown ARMS is superior to Chord, a well-known structured network, under various performance measures.

Keywords:

Object-based distributed systems; Distributed searching algorithm; Randomised structured network; Naming models

1. Introduction

Objet-Oriented Programming (OOP) model has become the most popular approach for the development of applications. Due to the increasing need to handle more and more sophisticated software, this has led to active research on the design of more efficient execution of the object-oriented programs in a distributed environment. The OOP model is based on a simple concept that objects could communicate with one another disregard their locations. To facilitate communication, every object can be referenced through a symbolic name. In an ideal situation, such names are location independent where they do not need to reveal their physical locations. While the location dependent name provides efficiency, the transparent name provides support for transparency, scalability, and reliability. Thus, current distributed systems largely implemented a location independent naming scheme [1, 2]. Such naming scheme, however, implies the requirement of a dynamic mechanism to map the name of an object to its corresponding physical address.

A fixed name server is commonly used to provide name translation for distributed systems due to its simplicity in design. However, such server has issues such as poor scalability, a possible single point of failure, and bottleneck in the system performance. Consequently, a peer model has been widely studied to provide scalable and decentralised name translation for large-scale distributed systems. In a peer model, every node takes the responsibility of answering messages that query the node's local content. The node also has to delegate a message to its neighbours if the node cannot answer the query message. An overlay search network is therefore required to allow peer nodes to establish virtual links between other nodes for query delegation. In general, there are two common approaches to construct a naming system that provides dynamic name translation. They are unstructured model and structured model based on the design of the overlay search network.

Structured search networks [3-6] facilitate load balancing and efficient query routing. However, the tightly controlled structure destroys the flexibility and the related objects are not necessarily located close-by. This is known as the locality issue as discussed in [7]. In addition, the structured search network also complicates the updating procedure due to nodes arrival or departure, and object migrations. On the other hand, unstructured search networks [8-10] imply neither an object placement policy nor an overlay structure. However, routing efficiency and scalability are the main weaknesses of unstructured networks.

Consequently, an Adaptive RandoMised Structured network termed ARMS is proposed in order to improve search efficiency and flexibility of existing search networks. ARMS is based on a randomised structured network [11-15] that provides the flexibility of neighbour selection. On the other hand, the proposed model utilises a distributed Ant Colony Optimisation algorithm (ACO) to improve path exploration. The design of the model is described in Section

978-1-4244-2096-4/08/$25.00 ©2008 IEEE

2 and the location protocol is presented in Section 3. Section 4 studies different kinds of heuristic information to improve search efficiency. Section 5 gives the performance analysis. Section 6 presents related research work in distributed search. Finally, Section 7 concludes the paper.

2. ARMS – Adaptive RandoMised Structured Network

ARMS is a distributed naming model that aims at providing efficient and scalable naming services for an object-based distributed system. The heart of the model is a randomised structured network. ARMS allows each node to choose its neighbours with greater flexibility when compared to a structured network. Such flexibility also provides a platform for adaptability. Structured networks limit the ability of a node to explore better routes due to their rigid organisation and preliminary routing. Because the state of a practical network is constantly changing and possibly chaotic, it is desired to allow each node to decide a route based on the current status of the network in order to achieve better utilization.

In contrast to other randomised structured network, ARMS uses an adaptive forwarding mechanism based on a distributed Ant Colony Optimisation algorithm in order to support neighbour exploitation and path exploration. Compared to RASTER [15], ARMS requires just a single message to update the forwarding table of nodes that reside on the best route between the initiator and the queried target by piggybacking updating information in the reply message.

The Ant Colony Optimisation algorithm (ACO) is a probabilistic technique for solving optimisation problems. In this application this is used to find the best path on a graph [16, 17]. In nature, ants select a path via a stochastic mechanism that is based on the values of the pheromone and heuristic information. In ARMS, the use of heuristic information is twofold. First of all, the constructive heuristics provides guidance to ants at the early stage of the search. Secondly, the heuristic information can be changed during computation in order to adapt to the changing state of the network. That is, it helps ants to choose alternative paths in case the good solutions known previously become obsolete due to the dynamic nature of the network. Such mechanism is vital to support path exploration in a dynamic environment. In addition, parallel ants are used initially in order to build a path quickly. Once such path is found, other ants are likely to follow the path as a result of pheromone attraction. Consequently, ARMS can effectively reduce the network traffic which is the key issue of flooding-based approaches while it still keeps the resilience of flooding.

3. Architecture of ARMS

We use a well known parallel approach called Chord [4] to demonstrate the concept of ARMS and its implementation in a practical structured system. This section describes the architecture of ARMS and its location protocol.

3.1. Network topology and forwarding table

Chord nodes are arranged in a ring topology as illustrated in Fig. 1. Each node has an identifier that is produced by a consistent hash function as shown in [4]. Such identifier is location-independent and thus it is transparent to the structure of the network. Every object is allocated to one of the keys on the ring space. Every node is responsible to only a subset of keys on the ring space in order to balance the state of the routing table across the nodes. The key of an object is stored in a node that has the identifier immediately followed that key. Such node is also termed the successor of the key [4]. In Fig. 1, for instance, N6 is the successor of key I5. The importance of the consistent hash function is its support for parallel naming assignment. The performance of the object oriented model relies on the efficiency of object communication and object creation. Profiling study has shown that object creation is concurrent and thus the parallel naming scheme is well suited to provide name binding for an object-based distributed system.

In Chord, a query is forwarded to the neighbour that has the closest node identifier to the object identifier or key at the ring space. The efficiency of query forwarding relies on the design of the forwarding table at every node. For m bits identifiers, each Chord node maintains an m-level forwarding table. The forwarding table stores a pointer termed *finger* that is pointing to the successor of the key:

$$(n+2^{k-1}) \quad 1 \le k \le m, \tag{1}$$

where n is the node identifier; k is the k^{th} level of the forwarding table; and m is the length of identifier.

The use of *fingers* is to accelerate the key search as the distance from the next forwarding node to the target node is at the most half the distance from the current node to the target. Thus the number of forwarding nodes necessary will be $O(logN)$ with high probability. Detailed proof can be found in [4].

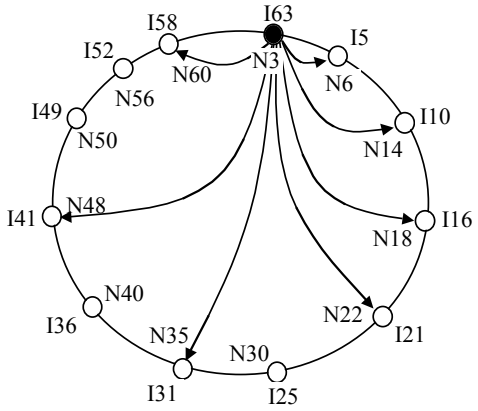

Figure 1. An illustration of Chord ring adapted from [4]

While there is only one node at each level of Chord's forwarding table, a node is allowed to add new neighbours to the k^{th} level of ARMS table when their node identifiers are belonged to the range of *2k-1* and *2k*. Although such design breaks the load balance of Chord's forwarding table, the flexibility of neighbour selection can improve the routing efficiency and latency resiliency.

3.2. Location algorithm

Upon name lookup, a node first checks whether the request key lies at its successor. If it is found, the query is forwarded to the successor. Otherwise, a node searches its forwarding table for the next node. If it is the initiator of the query, the node will forward the query to *q* neighbours in order to improve the speed of search. Or else, the node will propagate the query to one of its neighbours in order to reduce the network traffic. In contrast to finding the "closest" neighbour, as required in conventional structured networks, ARMS node looks for the "best" one amongst neighbours that are resided at the same level as the target in the forwarding table. The best forwarding neighbour is determined through the following probability process [16]:

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s^p)} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} \quad \text{if } C_{ij} \in N(s^p), \tag{2}$$

where τ_{ij} is the pheromone associated with the edge joining node *i* and *j*, $N(s^p)$ is the set of unvisited neighbours, the parameters α and β control the relative importance of the pheromone and the heuristic information η_{ij}.

Once the target is found, a reply is returned from the target through the forwarding path. Then, nodes on the path update their pheromone. Furthermore, in order to encourage the node to select different neighbours, the pheromone of the edge between the nodes *i* and *j* is locally updated once the query is forwarded to node *j*.

4. Constructive Heuristic Information

Heuristic information is used to guide the ants to choose their path to the target location. The choice of the heuristic information thereby affects the system performance. Consequently, we investigated four separate optimisation strategies, aiming at improving the efficiency of the execution, in this chapter. Furthermore, this study also shows the potential of ARMS for adapting to various performance specifications of a target application.

4.1. Optimising for proximity

The incompatibility between the virtual structure and the network organisation could result in a long travelling distance by the query message. Therefore, we modified the heuristic information of the DACO algorithm aimed at routing the query messages with the minimum travelling distance, e.g. the Manhattan distance. Instead of using the Manhattan distance, the link latency could be used as an alternative measurement. Furthermore, there are two ways to update the heuristic information: periodic updating or piggybacking in messages. Nonetheless, the piggybacking was used in our simulation since it requires fewer messages than the periodic updating approach.

4.2. Degree of connectivity

Objects sent or received messages varied in size as shown in our study. Intuitively, the objects communicated more frequently should have more knowledge of the addresses of other objects. As a consequence, the nodes that contain these popular objects are potential candidates for query forwarding although the workloads of these nodes may be higher than the rest. Here, we investigated the usefulness of the heuristic information based on the degree of connectivity for the lookup efficiency.

4.3. Optimising for load-balance

As shown in the profiling process, the amount of communications was not identical in every object and thus

hotspots may be developed in a structured overlay network, namely query hotspot and routing hotspot [18]. Query hotspots refer to popular objects which are frequently referenced by others. On the other hand, routing hotspots refer to objects that send query messages much more than others. In the former case, caching has been largely used to overcome query hotspots in the system. However, the routing hotspots are harder to manage in a structured overlay network due to the rigidness of the network.

ARMS reduces the number of hotspots in the system by two approaches: firstly, a query message can be transported via multiple routes taking the advantage of the ARMS flexible structure; secondly, a heuristic information that directs query forwarding towards lowly utilised nodes is used in order to evenly distribute query messages for processing at each node in the system.

5. Performance Analysis

We have conducted a series of simulation tests to study the impact of system parameters on the search efficiency and the scaling property of ARMS. The network traffics were produced from traces that were extracted from an execution of four object-oriented benchmark programs. Java [19] was chosen as the object-oriented language in this study due to its popularity and modernity. Originally developed with an objective to provide platform-independent applications, it is now more noted for its extensive use in networked applications. The benchmark reported in the paper is *AutoFocus (AF)* [20]. AutoFocus is a Java-based desktop search engine. It features in Cluster Maps to present the search results.

5.1. Computational model

The simulation system implemented two types of essential entities, computational nodes and active objects. A computation node is an abstraction of an execution unit, consisting of a data processing unit and a communication unit. An active object is responsible for reproducing the history of execution for a particular software object. For the sake of simplicity, each node ran exactly one object. Furthermore, a trace file contains only object communications profiled at runtime. We have assumed local computation follows a normalized distribution.

The design of active objects is based on a distributed object computational model termed the Actor model [21]. An actor performs a combination of the following actions in response to incoming messages:

the creation of new actors;

messages sent to other actors; and

updating of its state.

Furthermore, a 2-dimentsional grid was implemented because it is one of the most common structures used in cluster computing systems.

5.2. System parameters

The parameters investigated here are those that have direct or indirect impact on computation of the probability in Eq. (2) α and β. We tested several values for every parameter, as the rest being constant, in a 2D grid simulation network with the dimension of 40 by 40. The default value of the parameters was α=2 and β=10. Furthermore, four simulations for each configuration were performed to obtain mean values.

Here, we compared the performance of difference forms of heuristic information. A new scheme is also introduced, termed ARMS_AVG, where it simply takes the average value of the five forms of the heuristic information, namely lookup distance, proximity, degree of connectivity, spatial locality and load balance. The motivation is to study the potential of the use of a hybrid scheme in ARMS. Furthermore, Chord had been used as benchmarking in simulation tests. The value of the mean lookup length for each scheme is illustrated in Fig. 2 to Fig. 5 as the function of the network size. All schemes based on ARMS were superior to Chord due to the ability of discovering a shorter path between two nodes. The tests further showed that the heuristic based on the degree of connectivity could also lead to a relatively short lookup path on the virtual structure.

First of all, ARMS_LD and ARMS_PR showed the competitive values of the execution time as the average improvement in execution time for ARMS_LD and ARMS_PR was 21.2% and 21.6% over Chord. However, they approached the execution efficiency differently. While it incurs a long travelling distance, ARMS_LD produced the shortest lookup path in the virtual organisation as illustrated in Fig. 3. On the contrary, ARMS_PR emphases on routing the query with the minimum travelling distance on the physical distance. In other words, this approach is able to re-structure the virtual organisation based on the structure of the physical network. Hence, the value of distance per query (DPQ) was the lowest with ARMS_PR as illustrated in Fig.

4.

ARMS_DOC on the other hand, takes the advantage of the power-law distribution in the object communications, leading to a shorter lookup length and the faster execution time. Particularly, the execution time of ARMS_DOC was 14.2% quicker than the counterpart of Chord. The weakness of the scheme is its unbalanced distribution of workloads as illustrated in Fig. 5. As the result, the execution efficiency of ARMS_DOC is sensitive to the size of the object population as described later in this section.

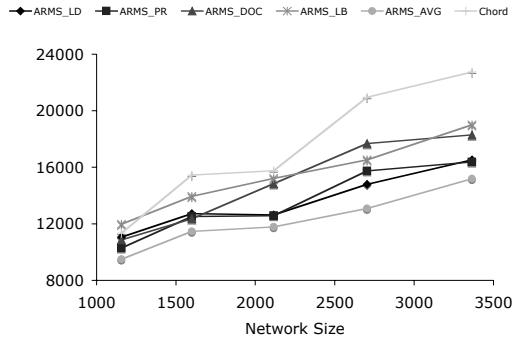

Figure 2. An illustration of the execution time for all schemes as a function of the network size through simulating the benchmark AutoFocus

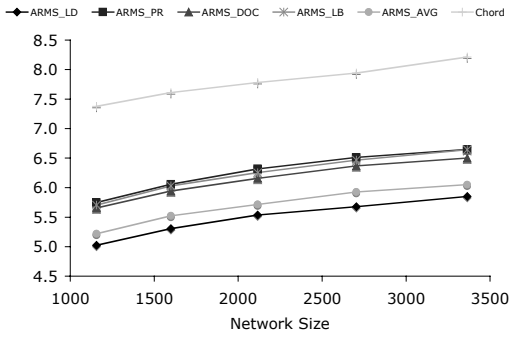

Figure 3. An illustration of the query path for all schemes as a function of the network size through simulating the benchmark AutoFocus

ARMS_LB is designed for maintaining equilibrium on workload across the system. As depicted in Fig. 5, the workload for ARMS_LB was lower than the other four schemes. However, the cost of balancing workload is the larger values of query length and travel distance as depicted in Fig. 3 and 4. Consequently, the execution time of ARMS_LB was the highest among the five schemes. Furthermore, it is noted that the workload of Chord was much lower than all schemes of ARMS because the use of parallel query messages in ARMS.

We further investigated the potential of a mixed scheme, ARMS_AVG, which takes the average of the five heuristic values. The tests showed that ARMS_AVG is superior to other schemes with respect to all performance measurements: query path, travel distance and load balance. As a result, the execution time ARMS_AVG was on average 26.8% faster than the execution time of Chord. The tests demonstrated that the ARMS algorithm is able to cooperate with inherently contradict parameters: at the beginning of the search, a node forwards a number of parallel query messages to its neighbours based on the computation of the probability. The computation is in turn determined by the quantity of heuristic information. Because of the decay coefficient, two messages are encouraged to take different routes that are controlled by separate heuristic information. Next, the quantity of the pheromone on the fastest route will be updated accordingly once the destination is reached. Consequently, the useful heuristic information is naturally selected through this process.

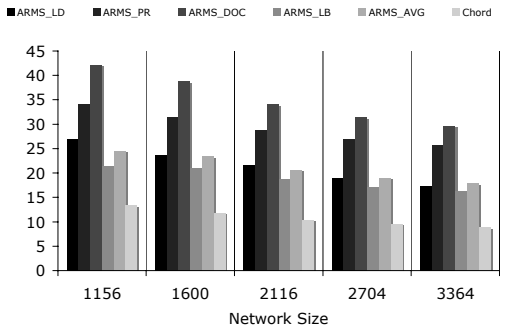

Figure 4. An illustration of the workload for all schemes as a function of the network size through simulating the benchmark AutoFocus

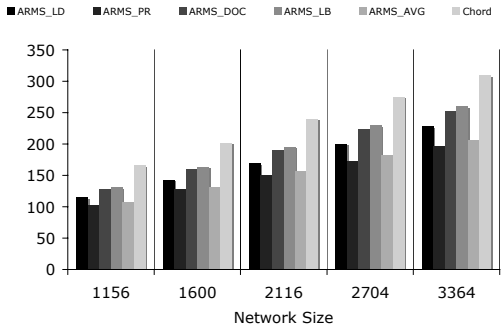

Figure5. An illustration of the travel distance for all schemes as the function of the network size through

simulating the benchmark AutoFocus

6. Related Work

Recent studies of location lookup for large-scale, dynamic networks focus on the use of distributed models, particularly in the area of peer-to-peer (P2P) file sharing systems [3-6, 8-10]. In a P2P system, each node has similar functionality and shares the responsibility of object binding and location lookup with other nodes. Because both control and data are distributed among the nodes, name services can be performed locally, leading to better scalability and efficiency. To find a node that is responsible for the particular symbolic name or key, the requesting node queries other nodes by sending messages. The two models - structured and unstructured search networks, for building distributed name translation system are discussed below.

The unstructured model implies neither a centralized control nor any structured data organization is deployed. Without the complete knowledge of the network structure, flooding is a typical approach to locate an object in the network. In this approach, a node queries all its neighbours within a certain radius [8] in order to locate the target object. This approach is extremely costly and leads to poor scalability and long response time.

Alternative protocols such as probabilistic flooding [9, 10], and random walkers [22] have been proposed to improve scalability and routing efficiency. In probabilistic flooding, a subset of the neighbours is selected to forward each query message. However, this model still gives poor performance due to the duplicated messages. Highly compressed data structures like Scalable Query Routing (SQR) [23] has been proposed to reduce the number of query messages being propagated to the neighbours.

Some better known structured networks are CAN (content addressable network) [3], Chord [4], Pastry [5] and Tapestry [6]. In these systems, namespace is structurally organized and tightly controlled. The namespace is uniformly assigned to each node in the network and hence the model ensures that the network load is balanced among the nodes. A different data placement policy may lead to a different routing scheme and thereby affects the performance of the location lookup process. Despite the popularity of the structured model, the model has several issues relating to the tightly controlled data placement, such as query hotspots and routing hotspots [7].

7. Conclusions

This paper introduces a distributed naming model, termed adaptive randomised structured search network (ARMS), for object based distributed systems. The model is based on a randomised structured model that provides the flexibility of neighbour selection. However, the model also incorporates adaptability by using a distributed Ant Colony Optimisation algorithm. Furthermore, the paper reports the performance of different heuristic information used to guide the search. In addition, more advanced approaches can be applied in replacement of the primitive averaging method in order to further improve the lookup efficiency. For instance, one can parameterise the heuristic information. Then, a decision tree would be used to control the importance of different kinds of the heuristic. Through this, the efficiency of the algorithm can be tuned based on the characteristic of the object communications.

References

[1] MSDN, ".NET Remoting," in Secondary.NET Remoting. Place Published: Microsoft, [url] http://msdn2.microsoft.com/en-us/webservices/aa740645.aspx.

[2] I. Sun Microsystems, "An Overview of RMI Application," in Secondary An Overview of RMI Application. Place Published: Publisher, [url] http://java.sun.com Access Date.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, "A Scalable Content-Addressable Network," presented at ACM SIGCOMM, 2001.

[4] I. Stoica, R. M. Karger, F. Kaashoek, and H. Balakrishnan, "Chord: A scalable Peer-to-Peer Lookup Service for Internet Applications," presented at ACM SIGCOMM' '01, 2001.

[5] S. Rowstron and P.Druschel, "Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems," presented at IFIP/ACM International Conference on Distributed Systems Platforms, 2001.

[6] B. Y. Zhao, J. Kubiatowicz, and A. Joseph, "Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing," University of California, Berkeley, Technical Report UCB/CSD-01-1141 2001.

[7] P. Keleher, B. Bhattacharjee, and B. Silaghi, "Are Virtualized Overlay Networks Too Much of a Good Thing," in *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS*, vol. 2429, *Lecture Notes in Computer Science*: Springer, 2002, pp. 225-231.

[8] "Gnutella website," in Secondary Gnutella website. Place Published, [url] http://www.gnutella.com/.
[9] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A Local Search Mechanism for Peer-to-Peer Networks," presented at ACM CIKM '02, 2002.
[10] V. V. Dimakopoulos and E. Pitoura, "Performance analysis of distributed search in open agent systems," presented at International of Parallel and Distributed Processing Symposium, 2003.
[11] K. Gummadi, R. Gummadi, S. Gribble, and S. Ratnasamy, "The impract of DHT routing geometry on resilience and proximity," presented at ACM SIGCOMM'03, 2003.
[12] G. S. Manku, M. Bawa, and P. Raghavan, "Symphony: Distributed hashing in a small world," presented at 4th USENIX Symposium on Internet Technologies and Systems, 2003.
[13] M. Castro, P. Drschel, Y. C. Hu, and A. I. T. Rowstron, "Topology-aware routing in structured peer-to-peer overlay networks," presented at International Workshop on Future Directions in Distributed Computing, 2003.
[14] H. Zhang, A. Goel, and R. Govindan, "Incrementally improving lookup latency in distributed hash table systems," presented at ACM SIGMETRICS 2003, 2003.
[15] C. C. Wang and K. Harfoush, "RASTER: a light-weight routing protocol to discover shortest overlay routes in randomized DHT systems," presented at 12th International Conference on Parallel and Distributed Systems, 2006.
[16] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization: artificial ants as a computational intelligence technique," Universite Libre de Bruxelles, Belgium 2006.
[17] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Politecnico di Milano 1991.
[18] S. Ratnasamy, S. Shenker, and I. Stoica, "Routing Algorithms for DHTs: Some Open Questions," in *Routing Algorithms for DHTs: Some Open Questions*, vol. 2429, *Lecture Notes in Computer Science*, 2002, pp. 45-52.
[19] J. Gosling, B. Joy, and G. Steele, *Java Language Specification*: Sun Microsystems, Inc, 1996.
[20] "AutoFocus website," in Secondary AutoFocus website. Place Published, [url] http://www.aduna-software.com/home/overview.view.
[21] C. Hewitt, P. Bishop, I. Greif, B. Smith, T. Matson, and R. Steiger, "Actor induction and meta-evaluation," presented at Proceedings of the 1st annual ACM SIGACT-SIGPLAN symposium on Principles of programming languages, 1973.
[22] Q. Lv, P. Cao, and E. Cohen, "Search and Replication in Unstructured Peer-to-Peer Networks," presented at ACM ICS '02, 2002.
[23] A. kumar, J. Xu, and E. W. Zegura, "Efficient and Scalable Query Routing for Unstructured Peer-to-Peer Networks," presented at INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies, 2005.