AN OPTIMAL HEALTH SERVICE DISCOVERY FOR UBIQUITOUS HEALTH SERVICE PROVISIONING

Moses O. Olaifa, Sunday O. Ojo, Tranos Zuva

Department of Computer Systems Engineering,
Faculty of Information and Communication Technology
Tshwane University of Technology, Pretoria, South Africa
{olaifamo, ojoso, zuvat}@tut.ac.za

ABSTRACT

Locating appropriate service and communicating it to the requester at the point of need within the shortest possible time is a key issue in ubiquitous health service provisioning. Failure to provide the health service at the exact time needed can result in erroneous diagnosis and treatment. Hence, there is a need for real time service discovery of requested services for prompt healthcare service provisioning. This paper proposes a peer-to-peer health service discovery solution that discovers and maintains a set of shortest service path to existing healthcare service types within a distributed health service network. We introduce an optimal service path cache (OSPC) for maintaining the shortest service paths provided by learning automata mapped to the super-nodes. The proposed solution was compared with existing methods and showed an improved performance in service discovery.

KEY WORDS

Peer-to-peer, service oriented computing, healthcare service, service discovery. ubiquitous.

1. Introduction

Timely response to healthcare service requests is a major goal of healthcare system [1] [2]. The provision of quality and real time health service delivery within the healthcare community is confronted with various challenges restricting the realization of the desired goal. This is due to highly distributed and heterogeneous nature of available healthcare services and resources. As a field comprising of time critical activities, health service provisioning requires adaptive and dynamic service-based approaches to address the challenges in the system. Moreover, the distributed services require coordinated approach in delivering appropriate and reliable services when and where they are required. For example, when a patient requires diagnosis and treatment from a physician other than his primary healthcare physician, there may be need to access the patient's medical record. In emergency cases, urgent decisions may be required to save the life of a patient and this will necessitate a real time access to the needed resources and services.

Different IT solutions have been explored and integrated into healthcare provisioning under the term electronic health (eHealth) [3] [4]. One of the solutions that build interacting services emerging from distributed and heterogeneous resources and applications is the service oriented computing (SOC) [5]. SOC depends on service oriented architecture (SOA), which integrates distributed and heterogeneous applications into a set of interoperable services [6]. For areas such as eHealth, SOA offers a broad model for implementing large scale enterprise applications which defines the interaction between the health service providers, requester and the discovery of the services the provider advertises by the requester. One of the major challenges in the area of SOA application in ehealth is real time health service discovery. As mentioned earlier, healthcare service request demands real time discovery of needed service to prevent unfortunate occurrences. This will enable an improved healthcare service delivery in diagnosis and treatment of patients.

This paper presents an intelligent real-time healthcare service discovery for a typical healthcare service networks. This research work introduces an approach that learns an optimal path to a service type that is requested by a patient. When subsequent requests are made for such service type, the system selects the service path with the

DOI: 10.2316/J.2014.216.815-0004

highest probability to the service. The remainder of this paper is arranged as follows. Section 2 describes few of the related work. Section 3 describes a typical peer to peer service discovery framework. Section 4 gives a description of our health service discovery system.

2. Related work

A typical peer-to-peer resource sharing network is made up of a large number of super-nodes [17]. The super-node manages the registration and access to a number of local computing resources linked to it. The super-node plays two major roles – resource producer and resource consumer. As a resource provider, it allows the implementation of other super-nodes using its local resources and as a resource consumer; it uses the local resources of other nodes to execute its tasks. When a resource request is forwarded to a super-node, it checks the information describing its local resources for the availability of the resource. If the resource is available, the search terminates, otherwise, the query is forwarded to the neighboring super-nodes. This process continues until the requested resource is found. Our system differs from this approach in that each super-node is mapped to a learning automaton that learns and provides the shortest service path to the different service types.

aring network is made up of a large number of super-nodes [17]. The super-node manages the registration and access to a number of local computing resources linked to it. The super-node plays two major roles – resource producer and resource consumer. As a resource provider, it allows the implementation of other super-nodes using its local resources and as a resource consumer; it uses the local resources of other nodes to execute its tasks. When a resource request is forwarded to a super-node, it checks the information describing its local resources for the availability of the resource. If the resource is available, the search terminates, otherwise, the query is forwarded to the neighboring super-nodes. This process continues until the requested resource is found. Our system differs from this approach in that each super-node is mapped to a learning automaton that learns and provides the shortest service path to the different service types.

Considering the increase in the size and wide range of computer networks and resource sharing systems, discovery of resources and services becomes important. The scale of deployment of a service discovery mechanism determines the design [7].Various search techniques are employed by service discovery mechanisms to discover services. Some of these search techniques include ant colony optimization [8], breadth-first search [9], hill-climbing search [10], iterative deepening [11], and probabilistic forwarding [12]

Different efforts have been made to improve service discovery in SOA based technologies such as web service, grid and cloud computing. Tao et al. [13] proposed a grid service discovery technique based on open shortest path first (OSPF) to address the challenges of large query response latencies and potentially overload in higher level service index servers. The approach used in this work combines the strength of both tree and flat architectures. The method reduces small size networks.

Banaei-Kashani et al [14] presented a web services peer-to-peer discovery service approach (WSPDS) based on a fully decentralized and discovery with semantic level service matching capability. To resolve the query forwarded by a peer, the network of WSPDS servents

mechanism determines the design [7].Various search techniques are employed by service discovery mechanisms to discover services. Some of these search techniques include ant colony optimization [8], breadth-first search [9], hill-climbing search [10], iterative deepening [11], and probabilistic forwarding [12]

Different efforts have been made to improve service discovery in SOA based technologies such as web service, grid and cloud computing. Tao et al. [13] proposed a grid service discovery technique based on open shortest path first (OSPF) to address the challenges of large query response latencies and potentially overload in higher level service index servers. The approach used in this work combines the strength of both tree and flat architectures. The method reduces small size networks.

Banaei-Kashani et al [14] presented a web services peer-to-peer discovery service approach (WSPDS) based on a fully decentralized and discovery with semantic level service matching capability. To resolve the query forwarded by a peer, the network of WSPDS servents work in collaboration. The servent comprises of the

communication and local engines. Every query forwarded by a user is received by the servent that checks the local site for matching services. If the service is discovered in the local site, the servent sends a response message back to the query originator. Otherwise, the servent collaborates with other servents to propagate the query based on probabilistic flooding dissemination mechanism. The approach uses TTL to restrict the dissemination of a query.

In his work, [15] introduced a heuristic algorithm for grid service discovery based on the concepts of agents. The service descriptions of available resources are advertised by the grid nodes that are mapped into agents. Each agent associated to a grid node manages an Agent Information Table (AIT) for recording resource description and advertises existing resources to neighboring agents. The agent employs flooding whenever a resource is not found in the local and adjacent agents. This approach is not suitable in dense networks due to the limiting factor in scalability.

In [16], a P2P based grid resource discovery system (MAAN) was presented. This approach is an extension of the Chord system and supports both single and multi-attribute range queries. The chord and MAAN handle the single attribute and multi-attribute query respectively. Every MAAN node represents an instance of the Chord system. The paper presented two approaches in handling the multi-attribute queries. The first approach divides the query into sub-queries that are issued within the attribute's proper space. The approach combines the results generated by the sub-queries to verify if there exist any resource that matches the attributes specified by the query. The second approach presents the complex query as a single query to search for a resource that meets the requirements stated in the query. This approach eliminates bottleneck problems but forwards queries to one neighbor at a time. This may create query loss in situations of node failure.

This work proposes a service discovery approach based on distributed learning automata. Our system differs from most of these approaches in that each super-node is mapped to a learning automaton that learns and provides the shortest service paths to the different service types.

3. Learning automata

The main goal of a learning automaton (LA) is to improve its performance by repetitive adjustment of its action probabilities through a reinforcement algorithm, thereby identifying an optimal action from a set of finite actions [18]. The LA selects an action at random from the action set based on the action probability vector. The environment accepts the selected action as input and responds with a consequence to the selected action. The action probability vector is updated using the learning algorithm. For more on the properties of learning automata, we refer the reader to [18-20].

The LA can be formally defined as a 6-turple $\{\phi, \alpha, \sigma, P, A, G\}$ where ϕ is a set of internal states, α represents the set of *n* possible actions, $\sigma = [0,1]$ is the set of responses presented by the environment to the LA, *P* is the probability vector over the set of states, *A* denotes the reinforcement algorithm for updating the action probability vector for each step n based on the environment response, and G is the output function $G : \phi \to \alpha$.

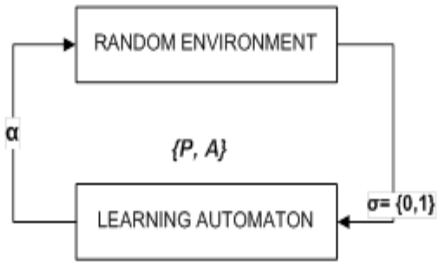

Figure 1: The learning automaton and the random environment.

The environment can be defined as a 3-tuple $\{\alpha, s, \sigma\}$ where $\alpha = \{\alpha_1, \alpha_2, \alpha_3, ..., \alpha_n\}$ is the finite set of input to the environment $s = \{s_1, s_2, s_3, ..., s_n\}$ represents the penalty probability vector, and σ is the set of output responses from the environment that are obtained by the LA. The responses are often binary $\{0,1\}$ where 0 and 1 denote penalty and reward respectively. If the set of environment responses is binary, the model is referred to as P-model. The other models are Q-model and S-model which refer to responses with finite collection of distinct symbols and an interval [0, 1], respectively.

According to [18], the behavior of a LA can generally be classified into expedient and optimal, which is described based on the average penalty. An average penalty for a LA with random action selection can be defined as

$$V_o = \frac{s_1 + s_2 + ... + s_k}{k} \qquad (1)$$

Assuming the LA selects an action α_i with probability P_i at step *t*, then the average penalty on $P(t)$ is

$$V(t) = \sum_{i=1}^{k} P_i(t) s_i \quad (2)$$

Therefore, a learning automaton is expedient if

$$\lim_{t\to\infty} E[V(t)] < V_o \quad (3)$$

That is, the average penalty is less than V_o

If the LA chooses the highest probability for an action associated with the minimum penalty, then it is said to be optimal. That is if,

$$s_i = \min\{s_i\} \quad (4)$$

then

$$\lim_{t\to\infty} E[V(t)] = s_i \quad (5)$$

4. Healthcare service search framework

Let $N \equiv \{n_1, n_2, n_3, ..., n_m\}$ represent the set of super-nodes, while $C \subseteq N * N$ represents the set of connections between the super nodes where $c_j \in C$ and $c_j : n_{i+1} \Rightarrow n_{i+m}$ is a connection path with sequence of nodes leading from n_{i+1} to n_{i+m}, then we assume $H'(N', C')$ represents the directed communication graph spanning from a connected undirected graph $H(N, C)$ of health service network. Let $S = \{S_k \mid 1 \le k \le m\}$ denote the set of advertised service types, where *m* is the number of service types. A learning automaton $A_i \in A$ is provided by the system for every super-node $n_i \in N$. We assume there exist k service paths with different lengths (number of traversed super-nodes) between the requesting nodes and providing nodes, where each service connection path $c_j \in C$ is associated with a probability from a probability vector P of all existing service connection paths for the service type. Based on the probability vector *P* containing the probability *p* of selecting each service path, the learning automaton A_i selects one of the service paths provided by the super-node. The environment is activated by the selected service path c_j which in response provides the consequence according to the consequence probability of the selected path. The consequence can either be a reward or penalty. The learning automaton uses a learning algorithm to update the service connection paths set probability vector *P* based on the consequence probability of the selected pathc_j. The automaton learns the optimal service path c_k that attracts the maximum probability. Furthermore, the learning automaton A_i maintains the set of optimal service paths for the different service types in the service path cache maintained by the super-node. The search mechanism selects the service path c_{ik} with the maximum probability that is closest (lowest number of traversed nodes) to subsequent service type request.

The learning automaton for our service network is defined as a 6-tuple $\{N, S, C, R, P, L\}$

- *N* is a set of nodes of service network *H*
- $S = \{S_k \mid 1 \le k \le m\}$ is the set of advertised service types in H
- $C \subseteq N \times N$ is a set of service connection paths where path $c_j(n_i, n_m)$ corresponds to action $\propto_{im}$ of automaton A_i.
- $R = \{a, b\}$ is a set of possible consequences for selected actions, where *b* denotes penalty and *a* denotes reward.
- P_k is a probability vector over a set of service connection paths for service type k such that, $P_{jk}(t)$ is the probability of selecting j^{th} connection path for service type k at step *t.*
- *L* is the learning algorithm automaton A_i uses to update the action set probability vector. If $P_{jk}(t)$ denotes the probability of selecting j^{th} path for service type k at step *t*, the probability is updated by the learning algorithm for a reward *a* as

$$P_{jk}(t+1) = \begin{cases} P_{jk}(t) + a[1 - P_{jk}(t)] & j = i \\ (1-a)P_{jk}(t) & \forall j \ne i \end{cases} \quad (6)$$

For penalty *b* the probability vector is updated as

$$P_{jk}(t+1) = \begin{cases} (1-b)P_{jk}(t) & j = i \\ \left(\frac{b}{r-1}\right) + (1-b)P_{jk}(t) & \forall j \ne i \end{cases} \quad (7)$$

Figure 2 describes the spanning graph $H' < N', C' >$ of graph $H < N, C >$ for a typical healthcare service network. A service request emanating from node *RN* for service type s_i located in *PN* can traverse different nodes within *H* to get to *PN*. As shown in the figure, there exist different service paths connecting *RN* and *PN*. Some of these service paths include:

$c_1 = RN \rightarrow n_2 \rightarrow n_3 \rightarrow n_4 \rightarrow n_8 \rightarrow PN$

$c_2 = RN \rightarrow n_{12} \rightarrow n_{13} \rightarrow n_{10} \rightarrow PN$

$c_3 = RN \rightarrow n_{12} \rightarrow n_6 \rightarrow n_5 \rightarrow n_3 \rightarrow n_4 \rightarrow n_8 \rightarrow PN$

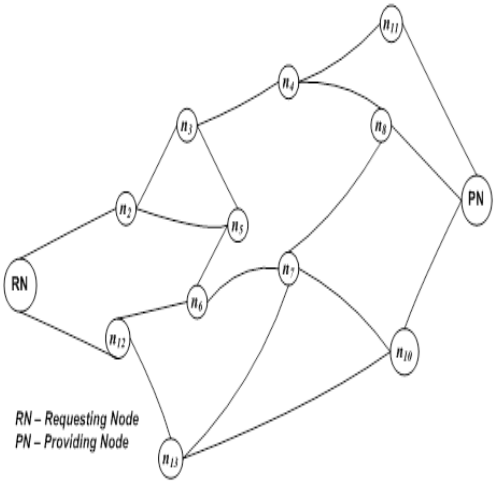

Figure 2: A typical communication graph for a service network.

c_1, c_2, c_3 traversed 5, 4 and 7 nodes respectively to remotely discover the requested service. c_2 attracts a higher probability than c_1 and c_3 because it has a shortest path to the required service.

We assume that a super-node $n_i$and a number of ordinary nodes make up a local health service community *h*. The super-node n_i manages an index server that holds information about existing services within the local community. When a user sends a query requesting for any health service, the query is directed to the local super-node to verify the availability of the service within the local community from the index server. If an advertised service that matches the requirements of the requester is discovered within the local health service community, a service-hit message is returned to the node where the query originated. If the requested service is not found within the local community, the super-node forwards the query to a set of neighboring super-nodes to search for any matching service in their respective local communities. Each neighboring super-node checks the set of service paths maintained by the OSPC within its community for the requested service. If the service path to the service is discovered in the node, the request is directed to the super-node with the requested service. Otherwise, the request is forwarded to the neighboring super-nodes.This process is repeated until the requested service is discovered. A service-hit message is returned with the service location to the requesting node once the matching service is found. Whenever a matching service is discovered, the connecting paths between the requesting node and provider nodes are maintained by the super-node of the requesting node for future decision making. To prevent the network load caused by sending of anomalous query within the network, we adopted the dynamic Time-To-Live (TTL) approach that decreases the TTL parameter each time a query is forwarded to a new super-node [21].

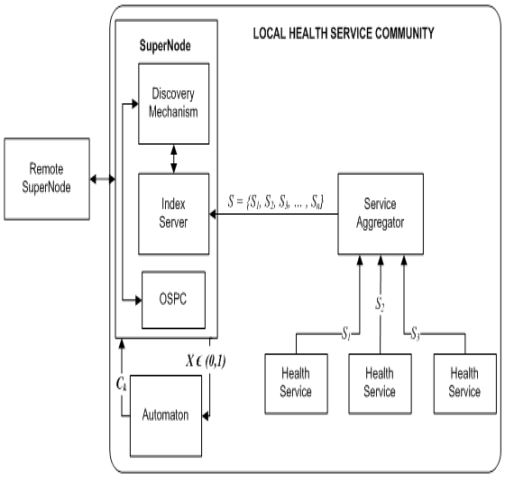

Figure 3: Local health service community framework.

Figure 3 describes a service network for local health service community. The super-node manages an index server that maintains the description of services provided by the nodes within the health service community and the optimal service path cache (OSPC) that is responsible for storing the shortest service paths to existing service types. The search mechanism processes the request and searches for services advertised by the index server that matches the service requested. The service aggregator monitors the services within the community and periodically updates the index server with current service status to guarantee the validity of the information provided by index server.

5. Evaluation and results

We evaluated our algorithm using DEVS-Suite simulator. We present different simulation results describing the performance of our proposed healthcare service search in a service network. The approach in this work is compared with both flood based and Genetic Ant Algorithm approaches. The experiment is carried out in an environment of 400, 500, 600, 700, 800 and 900 nodes respectively. The nodes which correspond to the vertices of a graph are randomly distributed, undirected with randomly distributed services among the nodes. For the experiment, we assume a service density of 0.05 in environments with nodes less than 500 and density of 0.06 otherwise. Our aim is to forward the service requests among the nodes and determine the length of the connection path by computing the number of traversed nodes as well as hit rate by the different algorithms. The number of traversed service nodes before and after update are presented in figure 4 and 5. It is shown figure 4 that the performance of the flooding approach is better than the Genetic ant algorithm and our approach. This is due to the fact that the flooding approach sends anomalous requests to the different nodes. The performance is followed by the GAA and finally our algorithm. Figure 5 presents an updated version after the selection and reinforcement by the automaton. Our approach records the least number of traversed nodes while the flooding approach records the highest traversed. Our approach presents the least traversed node because, for every subsequent request for a service type, our approach selects the shortest path to the service and directs the request to the providing node.

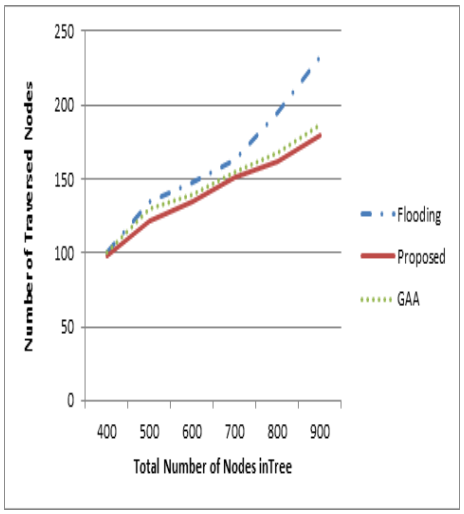

Figure 4: Average number of traversed nodes during the service search.

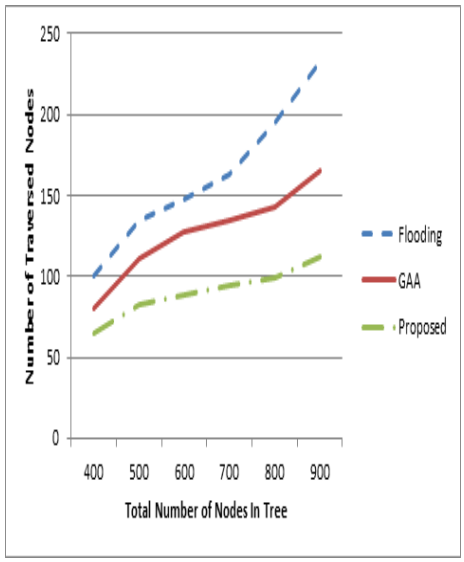

Figure 5: Average number of traversed node after update.

Figure 6 and 7 describes the hit rate of the three approaches before and after update. The hit rate defines the number of successful discoveries. Figure 6 shows that the hit rate between the three approaches are close before the update was made. In Figure 7, our approach shows a higher hit rate than the other two approaches due to the service density in the environment. In case of higher density, the three approaches may record hit rates at close range.

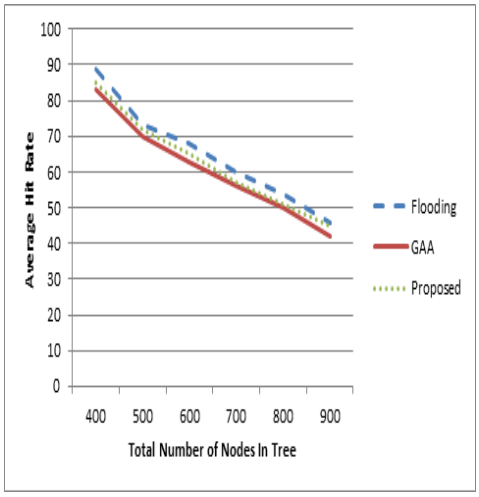

Figure 6: Average hit rate before update.

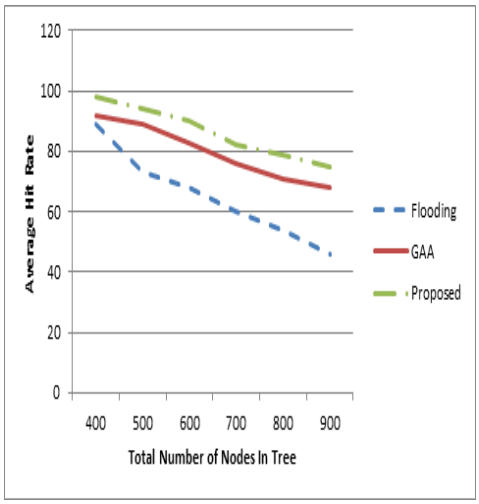

Figure 7: Average hit rate after update.

6. Conclusion

In this work, we presented a near real time healthcare service discovery approach that uses distributed learning automata. Our goal is to identify and keep track of the shortest paths to available service types. We have described the selection of connection paths from their sets and the update of their probability distributions to determine the optimal path. The experiment shows that our approach performs better than the flooding and GAA in subsequent request for a service type. For future works, the performance of the algorithm can be optimized to accommodate larger number of connection paths.

Reference

[1] Hameed, S.A., et al., *An efficient emergency, healthcare, and medical information system.* International Journals of Biometric and Bioinformatics (IJBB), 2008. **2**(5): p. 1-9.

[2] Coskun, N. and R. Erol, *An optimization model for locating and sizing emergency medical service stations.* Journal of medical systems, 2010. **34**(1): p. 43-49.

[3] Ball, M.J. and J. Lillis, *E-health: transforming the physician/patient relationship.* International journal of medical informatics, 2001. **61**(1): p. 1-10.

[4] Hardey, M., *'E-health': the internet and the transformation of patients into consumers and producers of health knowledge.* Information, Communication & Society, 2001. **4**(3): p. 388-405.

[5] Papazoglou, M.P. *Service-oriented computing: Concepts, characteristics and directions*. in *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*. 2003. IEEE.

[6] Cao, B.-Q., B. Li, and Q.-M. Xia, *A service-oriented QoS-assured and multi-agent cloud computing architecture*, in *Cloud Computing*2009, Springer. p. 644-649.

[7] Meshkova, E., et al., *A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks.* Computer Networks, 2008. **52**(11): p. 2097-2128.

[8] Dorigo, M., *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006, Proceedings*. Vol. 4150. 2006: Springer-Verlag New York Incorporated.

[9] Bader, D.A. and K. Madduri. *Designing multithreaded algorithms for breadth-first search and st-connectivity on the Cray MTA-2*. in *Parallel Processing, 2006. ICPP 2006. International Conference on*. 2006. IEEE.

[10] Gent, I.P. and T. Walsh. *Towards an understanding of hill-climbing procedures for SAT*. in *AAAI*. 1993. Citeseer.

[11] Lv, Q., et al. *Search and replication in unstructured peer-to-peer networks*. in *Proceedings of the 16th international conference on Supercomputing*. 2002. ACM.

[12] Liu, C. and J. Wu. *An optimal probabilistic forwarding protocolin delay tolerant networks*. in *Proceedings of the tenth ACM international symposium on Mobile ad hoc networking and computing*. 2009. ACM.

[13] Tao, Y., et al. *GNSD: a novel service discovery mechanism for grid environment*. in *Next Generation Web Services Practices, 2006. NWeSP 2006. International Conference on*. 2006. IEEE.

[14] Banaei-Kashani, F., C.-C. Chen, and C. Shahabi. *Wspds: Web services peer-to-peer discovery service*. in *Proceedings of the International Conference on Internet Computing*. 2004. Citeseer.

[15] Ding, S., J. Yuan, and L. Hu. *A heuristic algorithm for agent-based grid resource discovery*. in *e-Technology, e-Commerce and e-Service, 2005. EEE'05. Proceedings. The 2005 IEEE International Conference on*. 2005. IEEE.

[16] Cai, M., et al., *Maan: A multi-attribute addressable network for grid information services.* Journal of Grid Computing, 2004. **2**(1): p. 3-14.

[17] Beverly Yang, B. and H. Garcia-Molina. *Designing a super-peer network*. in *Data Engineering, 2003. Proceedings. 19th International Conference on*. 2003. IEEE.

[18] Narendra, K.S. and M. Thathachar, *Learning automata-a survey*. Systems, Man and Cybernetics, IEEE Transactions on, 1974(4): p. 323-334.

[19] Narendra, K. and M. Thathachar, *Learning automata: an introduction. 1989.* Printice-Hall, New York.

[20] Narendra, K.S. and M.A. Thathachar, *Learning automata: an introduction*2012: DoverPublications. com.

[21] McGeehan, J.E., et al. *Optical time-to-live decrementing and subsequent dropping of an optical packet*. in *Optical Fiber Communication Conference*. 2003. Optical Society of America.