MIRROR SITE ORGANIZATION ON PACKET SWITCHED NETWORKS USING A SOCIAL INSECT METAPHOR

P. Shi, A. N. Zincir-Heywood and M. I. Heywood
Faculty of Computer Science, Dalhousie University, Halifax NS, Canada
{ pshi@cs.dal.ca | zincir@cs.dal.ca | mheywood@cs.dal.ca }

Abstract

An approach making use of social insect metaphor and potential function clustering is employed to identify the connectivity pattern between clients and servers in order to ease the selection of mirror sites for application servers on computer networks.

Keywords: *Information technology and networking*

1. INTRODUCTION

The organization of data and services on distributed information systems has received interest from many perspectives over a long period of time, in particular distributed database design [1]. However, as indicated by Oates, Corne and Loader, this typically results in the problem being expressed as a single database that is geographically distributed [2]. The ensuing design emphasizes a very fine level of granularity – the allocation of data to specific files, their organization and the associated timing of operations. By doing so the dynamic properties of the network on which such systems are based is reduced. By taking such a view, the implicit assumption is that global knowledge of network load, configuration and routing is available. The approach taken by Oates et al. is therefore to consider the problem from a different perspective. Database servers and client nodes are considered to be distributed geographically, the variation in network parameters is considered to be the most important property, and the free parameter in this case is the client server interconnectivity as opposed to the movement of data between servers [2]. In order to address this problem, a matrix of communication times is used to express the time taken for a (client) request to be serviced (by a server). Such information is used to express server transaction times and usage profiles for clients and servers. The design problem is therefore to find the client-server connectivity pattern optimizing the overall performance of the distributed system. Such a problem is naturally a function of the network load profile and subject to change depending on usage profiles. In order to address this problem, various GA approaches were investigated, with a particular emphasis placed on the identification of appropriate crossover, mutation, selection and fitness functions [2, 3]. Bilchev and Olaffson investigate the same problem, but with a specific emphasis on the speed/ quality tradeoff between GA and greedy search techniques [4].

In both works, however, collection of the necessary network load profile information is not addressed. This is not a trivial problem as the sourcing of such information presents a global problem in itself [4]. Moreover, the algorithms employed require that reconfiguration be governed by a single centralized server. In this work, we therefore take a different approach. Firstly the collection of load profile information is conducted in a distributed manner. To do so we consider the problem from the server side. The objective is therefore to find the shortest path linking all clients to each server, where this problem is solved using a social insect metaphor [5]. Secondly, each server clusters the returned shortest path route with the objective of identifying potential sites for mirrors. On allocation of mirror sites, network load naturally changes and the process iterates.

In the following, section 2 summarizes the approach used, detailing the modifications used to define the Ant System in this setting, and how the clustering problem is solved. Results are presented in section 3 and conclusions made in section 4.

2. CLIENT-SERVER INTERCONNECTIVITY MODEL

As indicated above, the first objective is to identify the connectivity pattern between clients and servers. This problem is considered to be an example of the classic traveling sales man problem in which distance is now measured in terms of time as opposed to miles between cities. Sub-section 2.1 details how the original Ant System of Dorigo et al. is adapted for this purpose [5]. At this stage each server has collected all the information necessary to make a judgment regarding its usage profile. The second step is to cluster this profile, thus identifying the regions of the distributed system most likely to benefit from mirrored data. Sub-section 2.2 details how Potential Function clustering [6] facilitates this process.

2.1 Distributed collection of connectivity patterns

The Ant System of Dorgio et al. is a distributed optimization algorithm for performing a population-based search [5]. The social insect metaphor at the core of the approach has been shown to lend itself to several related problems, such as the packet routing problem (AntNet) [7] and the quadratic assignment problem [8]. Given that the central objective here, however, is to identify the shortest path linking client nodes to server nodes, without also optimizing routing strategies, we retain the original formulation of [5] in favor of the AntNet configuration [7]. Other instances of the algorithm, such as that for quadratic optimization [8] and graph organization [9] where not deemed applicable due to their utilization of global information that we are not able to infer in this application.

The basic motivation for such methods is that simple agents are able to perform complex tasks through interaction with the environment (measurement and payoff) and information previously left by other agents (positive feedback mechanism). In the context of this work this provides the basis using local information (order in which clients are visited) to infer global solutions (optimal order of visits) without recourse to a random walk (positive feedback).

The algorithm as a whole takes the following form. A (application) server is naturally aware of the most frequent users. This represents the set, *N*, of *K* (client) nodes for which a shortest path tour is required. Such a tour represents a fully connected graph (*N*, *E*) in which each of the *K* nodes may only be visited once, and *E* is the set of edges between nodes. The distance between any two nodes, d_{ij}, is expressed as the time a packet takes to travel between nodes *i* and *j*. The server initializes Ants on each of the *K* contributing nodes and trail intensity between nodes, t_{ij}, is reset. Each ant is then able to make a tour, subject to the following constraints,

- Each Ant maintains a tabu list, which keeps a record of the nodes visited on the current tour. This stops Ants from visiting the same node more than once, although passing through a node on route to a different destination is permitted;
- Prospective next destinations are selected probabilistically as a function of trail positive feedback, t_{ij}, and distance, d_{ij}, to the destination;

The routing tables used to support such information correspond to the Open Shortest Path First (OSPF) protocol, as is typically employed on network backbones. Thus, all the above information is readily available in practice (more will be said of this in the conclusion). Transition probabilities used to determine the k^{th} (forward) ant's next destination are defined by one of two conditions, depending on whether the an has previously visited the destination, $j \in \{N\text{-tabu}(k)\}$, in which case,

$$p_{ij}^{k}(t) = \frac{(t_{ij})^{\alpha}}{(d_{ij})^{\alpha} \sum_{k \in \{N-tabu(k)\}} (t_{ik})^{\alpha} / (d_{ik})^{\alpha}} \qquad (1)$$

Trail positive feedback factor, t_{ij}, expresses the popularity of a particular edge, E, and $1/ d_{ij}$ biases selection in favor of shortest routs.

In the case of a previously visited destination or a destination with probability below 0.0001, then $p_{ij}^{k}(t) = 0$.

Once ant *k* has completed its tabu list, then a return path is performed (backward ant) during which positive feedback is supplied, or

$$t_{ij}(t + 1) = \rho\, t_{ij}(t) + Q / L_k \qquad (2)$$

where Q and ρ are constants and L_k is the tour length.

Given the above definitions, the ant's follow the generic Ant Cycle algorithm, [5], interpreted as follows, in order to identify the shortest route interconnecting the clients associated with each application. That is, each application server provides a tabu list of clients they wish to improve service against. For a list of '*K*' nodes, *K* 'forward' ants are created, each with a unique starting node. This forces each ant to be initialized at one of the nodes on the server client list. Initial trail feedback, $t_{ij}(0)$, is set to small initial values and each ant chooses a path in conjunction with (1), node '*i*' is pushed into the *i*th ant's tabu list and ant '*i*' travels to node '*j*'.

Thereafter, if the '*k*'th ant is a forward ant, and its tabu list is complete it updates its total path duration and becomes a 'backward' ant. A forward ant with remaining destinations on its tabu list repeats the process for selecting a new destination. On reaching the next node record on the ant's tabu list, backward ants update the positive feedback parameter, (2) recalculate the probabilities of all connected edges (1) and rewind the tabu list one entry. When a backward ant encounters the start node, in addition to updating (2) and (1), it updates the corresponding server of the current shortest path and returns to the forward ant state. The process repeats until the server is satisfied that a best-case path has been identified – maximum number of ant cycles or ants all returning the same best-case route.

2.2 Profile clustering

As indicated above the second phase of the algorithm is to identify any natural organization, i.e. clustering, of the best-case route. Such clusters then define the location for mirror sites used by servers should client quality of service guarantees be sufficiently improved. To do so implies an assumption regarding the characteristics of the shortest route. Where it is certainly possible for a path to result in more clusters than is actually necessary, future work will investigate the significance of this. The clustering

algorithm used in this case is the Potential Function method [6], where this does not require additional *a priori* knowledge regarding the number of cluster centers. The clustering algorithm consists of four steps,

1. Identify the potential of each candidate point, as quantified by a suitable distance metric, with respect to all other points;
2. Select the point with highest potential;
3. Subtract the potential of the point identified at step two from the others;
4. Repeat on step two until end criteria is reached.

The metric at step one takes the form of a 'potential function', $P_t(x(j))$, or

$$P_t(x(j)) = \sum_{i=1}^{K} \exp\left(-\alpha \left\| x(i) - x(j) \right\|^2\right)$$

where $x(j)$ is the 'j'th time stamp associated with visiting the 'j'th client from the tabu list; t is the iteration of the algorithm; K is the number of client nodes and; α is the cluster radii constant.

Time stamps, $x(i)$, that are similar to the current candidate time, $x(j)$, therefore contribute most to the corresponding potential $P_t(x(j))$. Candidate times that have the most neighbors therefore receive the largest Potentials, as required at step two of the above process. Step three removes the influence of the current best case (largest) potential from all others within the same cluster. Thus, on identifying the winning (largest) potential, all potentials are reduced by a factor proportional to the distance from the current winning potential, $P_t(x^*(j))$, or for all $i \in \{1 \ldots K\}$,

$$P_{t+1}(x(i)) = P_t(x(i)) - P_t(x^*(j)) \exp\left(-\beta \left\| x(i) - x^*(j) \right\|^2\right)$$

where $P_{t+1}(x(i))$ is the updated potential at iteration $t + 1$ and; β ($< \alpha$) is the radii associated with the Potential decay process.

The process now iterates until the stop criteria is reached. This is expressed in terms of the ratio of the best-case Potential at the current time step and that of the first time step, or

IF Pt $(x^*(i)) > \gamma P_0(x^*(j))$ THEN create a new cluster
ELSE end

where γ takes the value 0.5 in the experiments reported later.

The assignment of time stamps, $x(i)$, to a cluster takes place by recording which 'j' resulted in the largest decrease during the Potential decay step. In the experimental study reported below, the clients corresponding to the cluster centers represent the candidate set of mirror sites. In practice, a further level of analysis could be employed in which the neighborhood of clients belonging to the same cluster is analyzed for mirror site location.

3. RESULTS

The authors have performed experiments on 5 different scenarios, Table-1, to test the approach described above. All experiments are employed by simulating a network, based on a topology as NTTnet (Japanese backbone – fig.1), with 55 nodes, and 71 bi-directional links. This implies that some nodes have high connectivity (many neighbours) and others have low connectivity (1 or 2 neighbours). Thus, the network also allows the simulation of worst-case scenarios (routes through low connectivity neighbourhoods) for more robust and reliable experiments.

Moreover, all of the experiments in this study are performed using *JavaSim-1.0* a component-based, compositional simulation environment [10]. For the purpose of network modeling and simulation, *JavaSim* provides a generalized packet switched network model. The model defines the generic structure of a node (either an end host or a router) and the generic network components, both of which can then be used as base classes to implement protocols across various layers. All simulations are run on a PC platform (700MHz, 256MB RAM, Windows 2000).

The following study is performed using bandwidth of l.5Mbps per link, and propagation delay of 5ms is used to simulate a pessimistic cases. Table-2 presents a summary of the scenarios after clustering and table-3 summarises the average values for throughput (in bps) and queue lengths (in number of packets/s) before and after the mirror sites were introduced.

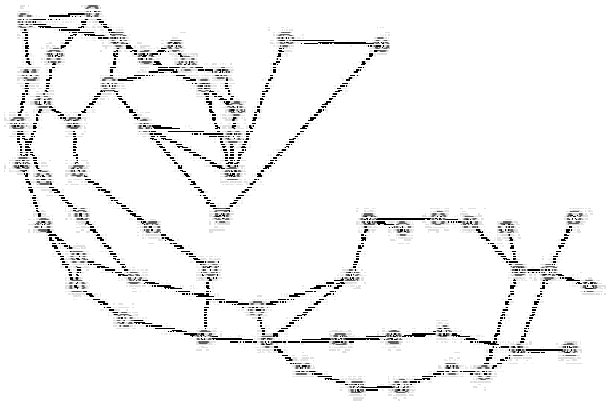

Fig.1: Japanese backbone - NTTNet

Table-1: The summary of scenarios used in this study

Scenarios	# of Nodes – candidates for mirror (s)	Candidate nodes
S1	12	49, 29, 27, 24, 34, 37, 19, 44, 40, 47
S2	18	35, 52, 54, 3, 8, 9, 33, 32, 27, 23, 24, 41, 39, 20, 46, 16, 11, 10
S3	20	1, 0, 52, 50, 53, 36, 33, 35, 20, 46, 16, 39, 41, 8, 9, 3, 24, 23, 32, 27
S4	25	50, 52, 0, 54, 53, 49, 1, 36, 3, 35, 8, 9, 10, 46, 20, 16, 11, 39, 41, 27, 32, 29, 33, 24, 23
S5	18	39, 20, 46, 9, 8, 3, 23, 24, 27, 32, 50, 52, 0, 53, 1, 36, 35, 33

Table-2: The results of clustering for potential mirror site(s)

Scenarios	Total # of Nodes – on the network	# of Nodes – candidates for mirror (s)	# of Clusters – After the Potential fun.
S1	55	12	3
S2	55	18	3
S3	55	20	4
S4	55	25	4
S5	55	18	4

Table-3: Throughput and queue length measurements on the network before and after the mirror sites are formed.

Scenarios	Averages - Before Mirror Sites		Averages - After Mirror Sites	
	Queue Length	Throughput ($x10^5$ bps)	Queue Length	Throughput ($x10^5$ bps)
S1	4	4.97	2.8	4.6
S2	7.7	5.45	5.58	4.95
S3	3.86	5.2	2.2	4.72
S4	8.36	5.26	5.21	4.84
S5	7.2	5.1	5.14	4.82

From the results of table-3 it is evident that as the number of nodes using an application increases and as their diversity (number of clusters) increases, the performance of the proposed approach improves. Creating mirror sites, using this approach, improves the average throughput by 10%, which in return improves the response time for the users. Moreover, this approach also improves the overall load distribution on the network. Thus, the average number of packets waiting in node queues (per second) improves by approximately 43% as the number of nodes and diversity increases.

4. CONCLUSION

In this study, the authors have proposed and showed an approach making use of social insect metaphor and potential function clustering. These are employed to identify the connectivity pattern between clients and servers in order to ease the selection of mirror sites for application servers on computer networks. Experimental results show that the method works well on the scenarios introduced here. However, research work continues to test and optimize the method where the clusters are not distinct but more widely distributed (in other words, random client patterns). Furthermore, comparisons with other work are envisaged.

ACKNOWLEDGEMENTS

A. Nur Zincir-Heywood and Malcolm I. Heywood gratefully acknowledge the support of Individual Research Grants from the Natural Sciences and Engineering Research Council of Canada.

REFERENCES

[1] S.T. March, S. Rho (1995). Allocating Data and Operations to Nodes in Distributed Database Design. IEEE Transactions on Knowlegde and Data Engineering, 7(2): 305-317.

[2] M. J. Oates, D. Corne, R. Loader (1998). Investigating Evolutionary Approaches for Self-Adaption in Large Distributed Databases. IEEE International Conference on Evolutionary Computation. pp 452-457.

[3] M. J. Oates, D. Corne. (2000) Exploring Evolutionary Approaches to Distributed Database Management. In D. Corne et al. Telecommunications Optimization: Heuristic and Adaptive Techniques. Chichester, UK: John Wiley and Sons. ISBN 0-471-98855-3.

[4] G. Bilchev, S. Olafsson. (2000) Adaptive Demand-based Heuristics for Traffic Reduction in Distributed Information Systems. In D. Corne et al. Telecommunications Optimization: Heuristic and Adaptive Techniques. Chichester, UK: John Wiley and Sons. ISBN 0-471-98855-3.

[5] M. Dorigo, V. Maniezzo, A. Colorni. (1996) Ant System: Optimization by a Colony of Cooperating Agents. IEEE Transactions on Systems, Man and Cybernetics – Part B. 26(1): 29-41.

[6] S.L. Chiu. (1994) Fuzzy Model Identification based on Cluster Estimation. Journal of Intelligent and Fuzzy Systems. 2: 267-278.

[7] G. D. Caro, M. Dorigo. (1998) AntNet: Distributed Stigmergetic Control for Communications Networks. Journal of Artificial Intelligence Research. 9: 317-365.

[8] V. Maniezzo, A. Colorni. (1999) The Ant System Applied to the Quadratic Assignment Problem. IEEE Transactions on Knowledge and Data Engineering. 11(5): 769-778.

[9] P. Kuntz, D. Snyers. (1999) New Results on an Ant-Based Heuristic for highlighting he organization of large graphs. Proceedings of the 1999 IEEE Congress on Evolutionary Computation. pp 1451-1457.

[10] DRCL, OHIO University, (2002) DRCL JavaSim, http://javasim.cs.uiuc.edu/index.html