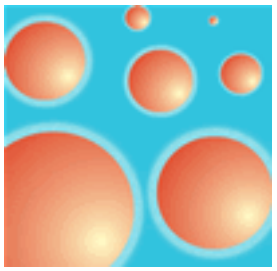
Crossroads
The ACM Student Magazine



An Interaction Model for Mobile Agent Services using Social Networks

by ***Vishakh***, ***Nicholas Urrea***, ***Tadashi Nakano***, and ***Tatsuya Suda***

Introduction

Computer networks have been experiencing dramatic changes in their nature over the past few years. Traditionally, network services have been provided through simple interactions between a service provider and service consumers (e.g., web server and clients). For example, CNN's web server provides its clients with news through HTTP (Hyper Text Transfer Protocol). Recent distributed networks, on the other hand, provide network services in a more complex manner involving a greater degree of interaction among service components [**1**]. Examples of this include mobile agent services [**2**, **6**, **8**] where a number of virtual software entities, called agents, migrate over networks to interact with each other and provide network services.

Figure 1 shows a simple example of mobile agent services, where agents interact with each other and collectively provide network services to a user. For example, the user, Sergio, has just awakened and walks up to his computer to perform his morning ritual of reading a summary of the world's news. His computer, which is connected to a mobile agent network, broadcasts a request for a news summary. A News Collector agent receives the request and migrates to Sergio's computer to provide him with his

desired service. The collector has been designed to gather news from different sources and summarize them. It then broadcasts a request for news services. News agents in the vicinity, which have been released by organizations such as the BBC and CNN, migrate to Sergio's computer. Sergio pays the collector for its services with "energy", akin to a monetary transaction he might make with a newsstand owner. Energy here is a commodity that users and agents use to purchase services. The collector in turn pays the news agents with energy. The news agents provide the collector with the latest news and the collector, in turn, provides a summary to Sergio. After this, Sergio catches up on the headlines over a mug of coffee, while the agents that served him move on to other locations where they are needed. If these agents collect energy above a defined threshold, then they can replicate to make more copies of themselves available on the network. Conversely, if they run out of energy, they die.

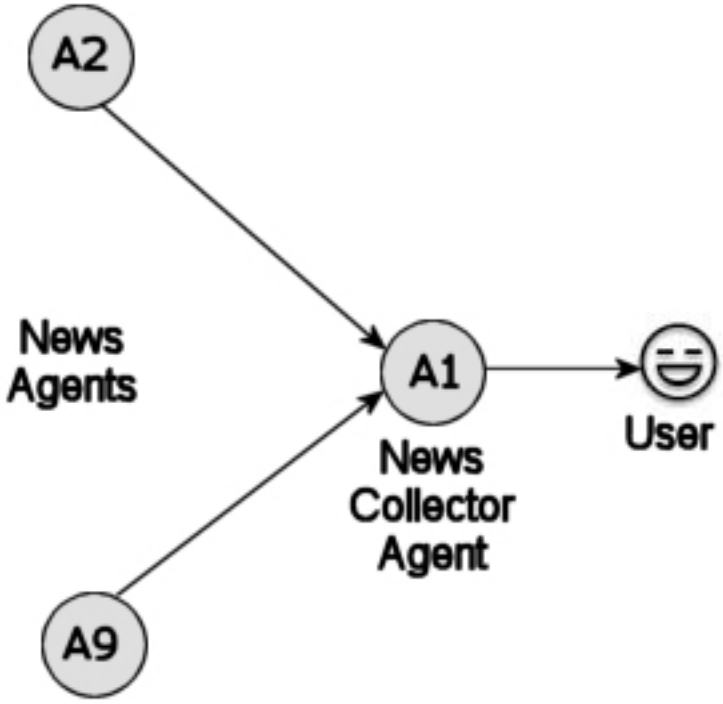


Figure 1: Sergio interacting with service agents on his computer.

Mobile agent services have several advantages [3] due to their decentralized nature:

- **Scalability:** They do not suffer from performance bottlenecks. In centralized systems, bottlenecks in computational power and bandwidth occur at the server side as the number of clients increases. These bottlenecks can only be overcome through hardware and software upgrades, often at substantial prices. Mobile agent services, however, are decentralized and due to this, the services that

they implement can be scaled up and down cheaply and easily.

- **Adaptability:** They handle varying user demand through adaptive behavior [5]. When demand fluctuates, agents replicate or delete themselves, thus adapting to the new conditions. Central servers, on the other hand, are set up according to a priori estimates of user demand. Due to their static nature, resources are either wasted or are proven inadequate in the face of changing demand. Agents also display spatial adaptation through migration. Even if user concentrations across a network fluctuate with time, agents migrate in a manner that ensures that their distribution is near-optimal at any given time.
- **Availability:** They handle network failures. Since agents do not rely on centralized control, even when some agents fail due to network disruptions, other agents can provide continuous service. In centralized systems, once central servers go down or become inaccessible, users are unable to receive network services.

Mobile agent services are being used increasingly due to the advantages they offer. We seek to simplify their study through our generic "N-services" model, which abstracts any given set of interacting mobile agent services. General studies of the nature of these services using our model will lead to their optimization. The term "N-services" refers to the fact that the model can be adapted to model any number of interacting mobile agent services. In our model, agents represent abstract entities that provide and consume services. In the model, agents interact with each other through a referral network, which is constructed by representing agents as nodes and adding links between the agents that consider each other as "friends." Referral networks are social networks in which job requests generated by agents are fulfilled by either their friends or their friends' referrals [10]. A social network consists of a group of people connected through various relationships, for example, friendships and acquaintances. A real-world example of a referral network would be that of a group of friends. If a person in the network needs to hire someone for a job, then they ask their friends to recommend someone with the requisite skills.

When an agent requires a particular service, it first looks to its friends. If one of its friends is ready to supply the service, then the agent is immediately satisfied. If not, its friends turn to their friends and so on until an agent that is ready is found. This information is propagated back to the original agent as a referral. When a referred agent is chosen to provide the service, it becomes a friend of the original agent. Thus, relationships between agents eventually change so that the resultant network links

them in a way that is conducive for each agent to quickly and reliably find other agents that provide services that they need.

We implemented the N-services model through computer simulations. In this paper, we present insights developed into the growth and formation of a referral network of agents and how efficiently services were produced through one of our simulations.

The rest of the article is organized as follows: The next section describes the N-services model. In it, we go though the procedure involved in the working of the model in a step-wise manner. In the following section, we list the simulation carried out to verify the viability of our model. It lists the simulation configuration and evaluation metrics used. It then describes the results and their implications for our model. Finally, we provide some concluding remarks about the model.

The N-services Model

The N-services model uses a referral network to define agent relationships and behavior. In it, the concept of energy is used as currency. When an agent receives a service from another, it pays the other agent with energy. This is analogous to a person paying a store for a service such as a haircut. Agents receive energy from other agents, depending on services they provide. They then use this energy for behaviors such as replication and migration. This allows agent behavior to be implemented with simplicity and also for efficient resource allocation. Agents that are not useful do not provide services that are not in demand and eventually die out as their energy is expended in getting services. Other agents that provide heavily-demanded services replicate more since they constantly receive energy. In this way, agent populations are regulated by supply and demand in an elegant, decentralized manner. We now list the steps involved in creating and running such a network using our model.

Service Interactions

In our model, agents implementing services need other services according to defined interaction probabilities. **Figure 2** shows an example of three services that interact with each other. In this example, agents that implement S1 would require services S2 and S3 with a probability of 0.5. Agents implementing service S3 would need S2 with probability 0.1 and S1 with 0.3. Finally, agents implementing service S2 would need S1 with a probability of 0.8 and S3 with 0.2.

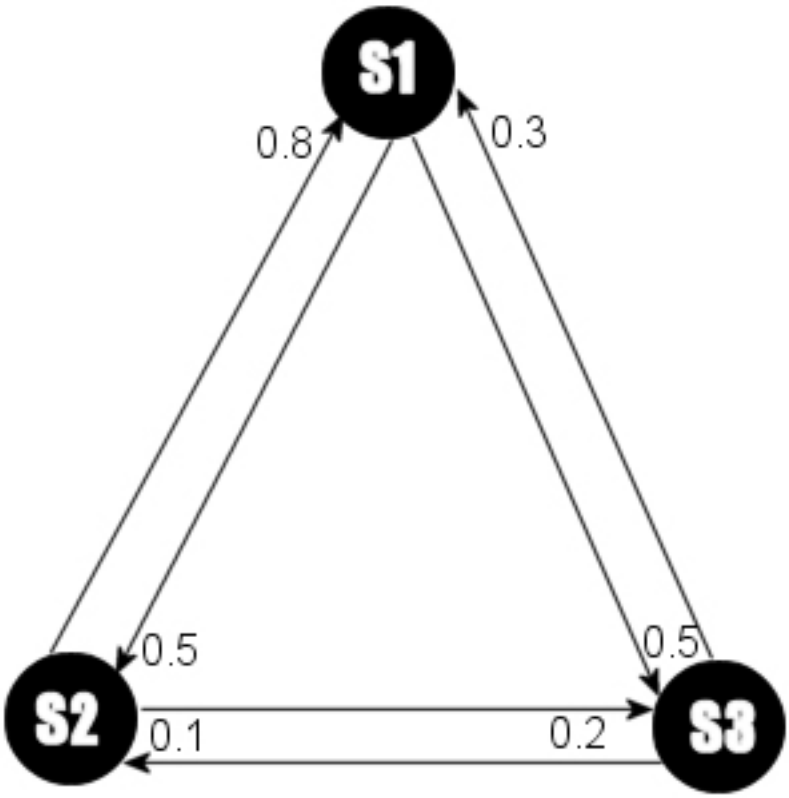


Figure 2: Three interacting services.

Agents in our model can always be found in one of three states. Agents in the "idle" state are not involved in an exchange of services with anyone. Those in the "providing" state provide a service to another agent at that instant. Finally, agents in the "receiving" state are in the process of receiving a service from another agent. Only idle agents can use interaction probabilities to look for and then receive a service.

Creating the Referral Network

The initial referral network is populated using pre-defined agents. An example of this can be seen in **Figure 3**. A referral network with four agents implementing three different services is shown. Friendships represented by links in the diagram are also defined. Note that links are unidirectional, meaning that friendship in the model is not reciprocal. Initially, all agents are idle. Each agent is allowed a fixed number of friends in our model. Thus, while an agent's friends can change, their numbers cannot. Each agent's friends are randomly allocated at the beginning for simplicity. Initially, the status of every agent is set to "idle."

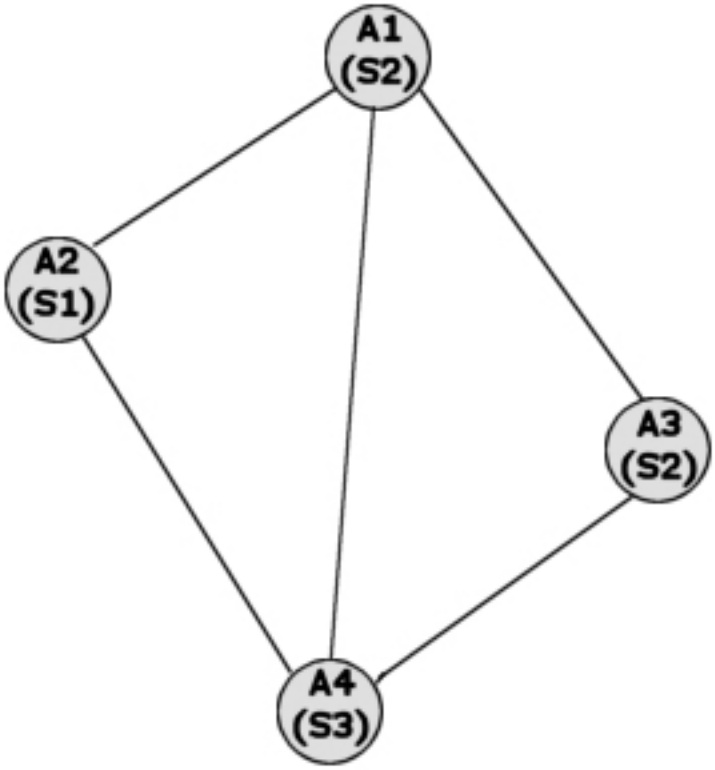


Figure 3: An initial referral network.

Requesting Services

Idle agents are checked to see whether they require a service using interaction probabilities. Once a service is chosen, the next task is to look for another agent from which to get it. The agent which creates a service request, called the service seeking agent (SS), forwards the request to its friends. This request is propagated along the network by an agent to its friends until an idle agent providing the service, a service-providing agent (SP), is found. When the request reaches an agent that is able to fulfill it, a referral is sent back with this agent's ID to the SS. After an SS makes a service request, it might get multiple referrals. The SS uses a distance metric, currently the number of links to the referred agent, to pick the best option. Thus, this referral is used to pick the SP. If the SP is already a friend of the SS, then the network remains unaltered. If not, the SS randomly removes one of its links and creates a link to the SP.

Figure 4 shows a small section of a putative referral network before and after a service request was satisfied. Initially, Agent A1 implements service S1 and is connected to A2 and A4, which implement services S2 and S4 respectively. A1 desires S3 and broadcasts a request for it. Its friends don't implement S3, but A2's friend A3 does. A2 sends back a referral to A1, which then receives the service from A3. It also

adds A3 as a friend. If adding A3 takes the number of A1's friends beyond a fixed allowable level, then one of A1's friends is randomly deleted.

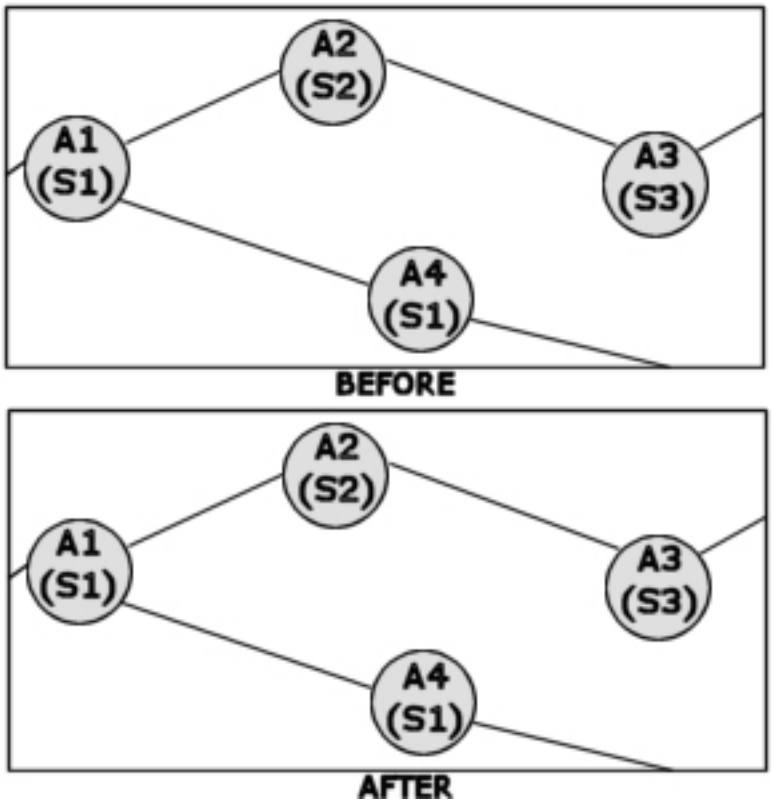


Figure 4: Fulfillment of a service request.

Modeling Agent Behavior

Agents in our model are assumed to possess the basic behaviors of migration, replication, and death. The N-services model implements these behaviors simply in terms of link (relationship) and node (agent) addition and removal. Replication is simply the addition of a node. When an agent is idle and possesses more than a pre-defined amount of energy (the Replication Threshold), it creates a copy of itself and gives it half of its energy. When an agent runs out of energy and possesses less than a pre-defined amount of energy (the Death Threshold), it is removed from the network, that is, it dies and the corresponding node is deleted.

Migration is relatively more complex than replication and death. Agents have no inherent spatial properties in the N-services model. In intuitive terms, agents migrate by moving over network platforms (the physical sites of which the network consists). In our model, platforms are also represented as agents that provide other agents with computing resources such as CPU time and memory. If an agent is using a platform service, then it means that the agent resides on that platform. Similarly, if the agent

starts using the services of another platform, then it means the agent has now migrated to that platform.

Simulation

We now discuss one of our simulations to demonstrate the N-services model and the following metrics to evaluate the simulation results:

- **The clustering-coefficient** of an agent is the ratio of the number of friends' friends that are the agent's friends to the total number of the agent's friends. This coefficient is a measure of clustering, which is the tendency of agents to form densely-connected cliques within referral networks. This usually happens when agents of services that need each other with high probability form links with each other. In such a case, the friends of an agent's friends will most likely be the agent's friends as well. **Figure 5** shows an example of a cluster. A high clustering-coefficient is generally desirable as agents that need each would be in close proximity and would have little trouble finding service providers.

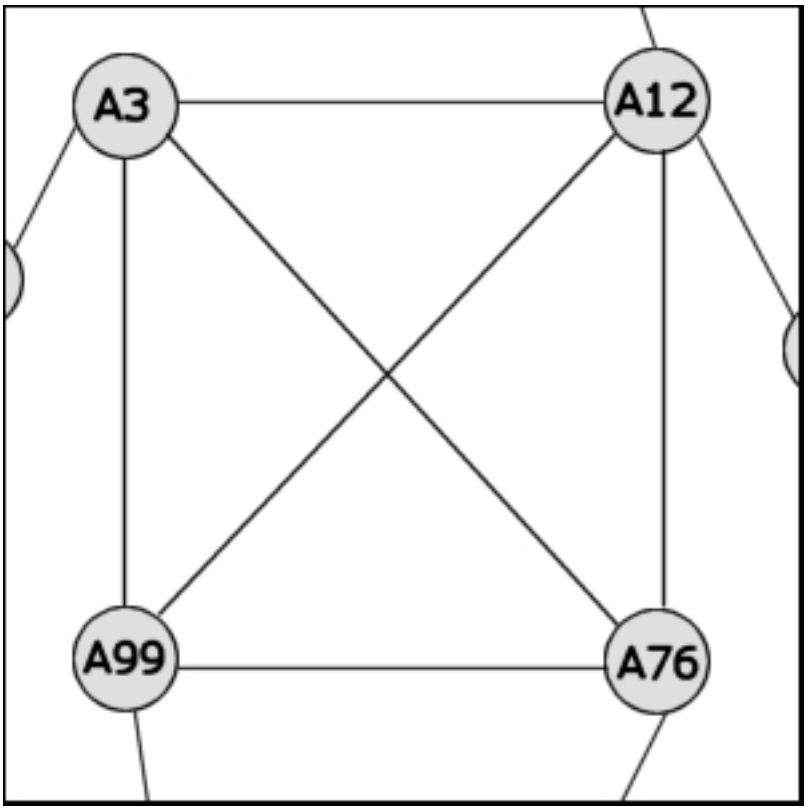


Figure 5: A Cluster.

- **Efficiency** is the ratio of service requests satisfied to the total number of requests made by all agents. An efficiency of 1 implies that all service requests made by every agent through the simulation were successfully fulfilled. A high degree of efficiency is the desirable outcome of our simulations.

A total of ten services were defined for the simulation and the interaction probabilities between them were randomly determined. The simulation was then run with the following parameters:

Parameter Name	Description	Value
PARAM_ITERATIONS	Number of iterations the simulation was run for.	1000
PARAM_TTL	Depth of the recursive search for service providers.	10
PARAM_MAX_FRIENDS_ALLOWED	Maximum number of friends each agent was allowed to have.	4
PARAM_SERVICE_DURATION	Number of iterations it took for agents to provide services to others.	1
PARAM_DEATH_THRESHOLD	If agents possessed less than this amount of energy, they died.	0
PARAM_REPRODUCTION_THRESHOLD	If agents possessed more than this amount of energy, they replicated.	20

One thousand iterations were found to be enough to spot emergent trends, while a value of 10 for PARAM_TTL allowed a deep search without a noticeable degradation in performance. The other parameters were chosen in an arbitrary manner. Most simulations produce the same trends as long as the initial parameters do not diverge greatly from the ones supplied above.

Figure 6 shows the change of population over the 1000 iterations. The population initially rises and then stabilizes with slight variances at around 120. This is because the initial agents posses energy much in excess of the reproduction threshold. They expend the excess energy through replication until the condition finally changes around the 100^{th} iteration.

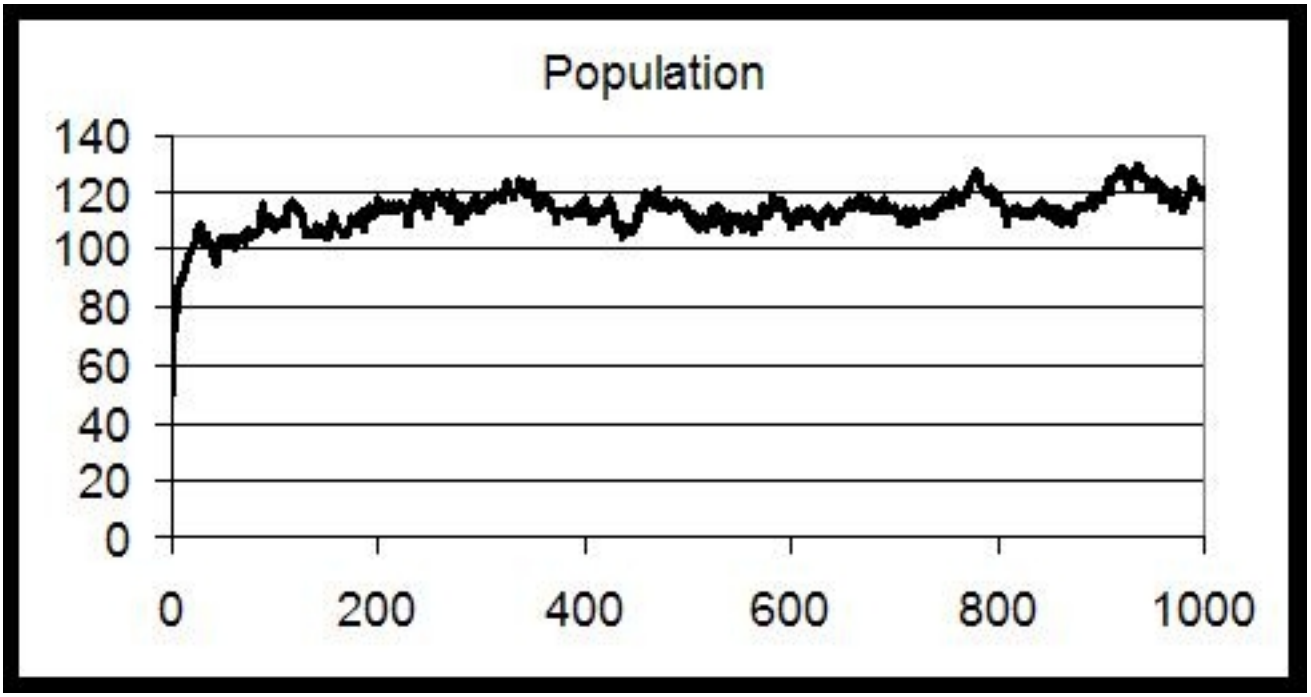


Figure 6: Change in population with time.

Figure 7 shows the change in clustering coefficient over the simulation. In the beginning, links between agents were randomly assigned, so very little correspondence existed between the needs and actual friendships of the agents. As the simulation progressed, service interactions led to more realistic friendships and agents that were of mutual use increasingly tended to become friends. This led to a rise in clustering. However, newly-spawned agents were also assigned random links through the whole simulation and this acted as a counterbalance to increased tendency to cluster. As a result, clustering levels stabilized by the 600th iteration.

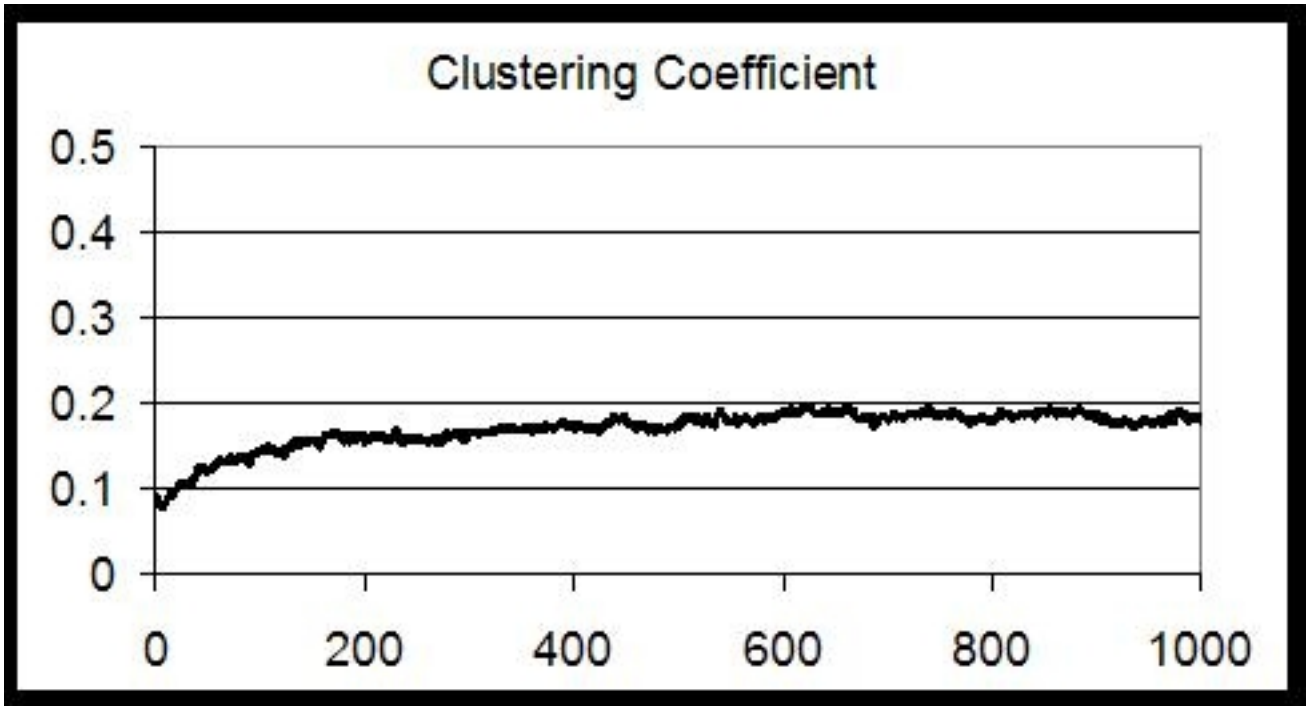


Figure 7: Change in Clustering-coefficient with time.

Figure 8 shows the change in efficiency over the course of the simulation. There is a continual decrease in efficiency before a tapering off close to a value of 0.4 starting at around the 600th iteration. It can be seen that efficiency and clustering are intimately connected here. A high level of clustering would have been beneficial had the services been strongly mutually dependent. In such a case, efficiency would have increased with clustering levels. However, service interaction probabilities in this simulation were randomly defined and there were very few concrete relationships between services. On the contrary, each service type's dependence was scattered over all the others instead of a select few. As such, clustering was actually a hindrance for successful service satisfaction since agents that looked for service providers could not find them easily within their own clusters. After the 600th iteration, when clustering levels stabilized, efficiency also started converging to 0.38.

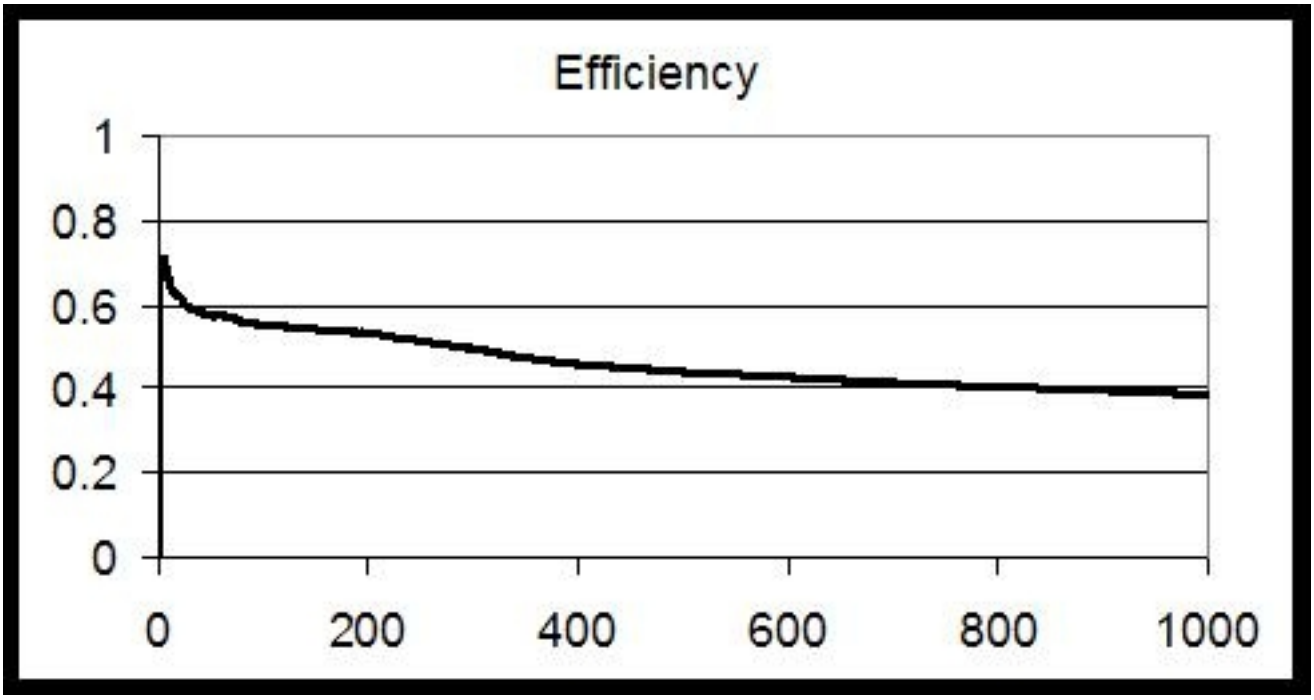


Figure 8: Change in efficiency with time.

These simulation results have important implications for our model. The results underscore the importance of an effective friend-allocation policy. Currently, initial friendships are randomly assigned, leading to sub-optimal performance. With a more effective policy, such as friend-selection for new agents based on the type of their parents′ friends, higher efficiency levels can be obtained. Service interactions also have to be designed in a manner that encourages strong mutual relationships between services. This would certainly lead to beneficial clustering. Finally, initial energy levels have an important effect on the whole simulation as they dictate the size of agent population. A more careful approach must be taken towards allocating initial energy levels to agents so that they are more reflective of the desired scale of the resultant mobile agent network.

Conclusion

There is much scope for future work. The model itself operates under very simple assumptions. Attempts have to be made to simulate real-world mobile agent services using the model. This will undoubtedly open new avenues and expose deficiencies in the current formulation. More metrics can also be used, such as expertise and sociability levels in [**11**] to investigate the model more thoroughly. Several features are already under consideration, such as allocation of random links based on some spatial consideration for more realistic effects and an auction-based model for finding service providers [**7**].

We hope that investigation of mobile agent service systems using our model will lead to fruitful insights into their working. The generic nature of our model makes it easy to adapt it to any set of mobile agent services. Through the study of a variety of such systems, we hope to provide insights into and guidelines for the design of future mobile agent services.

References

1

Carzaniga, A., Picco, G. P., and Vigna, G. (1997). Designing distributing applications with mobile code paradigms. In *Proceedings of the 19th international conference on Software Engineering*, ACM Press, 22-32.

2

Du, T. C., Li, E. Y., and Chang, A. (2003). Mobile agents in distributed network management. *Communications of the ACM*, 46(7): 127-132.

3

Hagimont, D., and Ismail, L. (1999). A performance evaluation of the mobile agent paradigm. In *Proceedings of the 14th ACM/SIGPLAN Conference on OOPSLA '99.* ACM/SIGPLAN, 306-313.

4

Kautz, H., Selman, B., and Shah, M. (1997). Referral Web: Combining Social Networks and Collaborative Filtering. *Communications of the ACM*, 40(3): 63-65.

5

Lerman, K., and Galstyan, A. (2003). Agent memory and adaptation in multi-agent systems. In *Proceedings of the 2nd Internaional Joint Conference on AAMS '03,* ACM Press, 797-803.

6

Suda, T., Itao, T., and Matsuo, M. (2002). *The Bio-Networking Architecture: The Biologically Inspired Approach to the Design and Scalable, Adaptive, and Survivable/Available Network Applications* In K. Park (Ed.) The Internet as a Large-Scale Complex System, Princeton University Press,.

7

Vulkan, N., and Jennings, N. R. (2000). Efficient Mechanisms for the Supply of Services in MultiAgent Environments. Decision Support Systems, 28: 5-19.

8

Wang, M., and Suda, T. (2001). The Bio-Networking Architecture: A Biologically Inspired Approach to the Design, Scalable, Adaptive, and Survivable/Available Network Applications. In *Proceedings of the 1st IEEE Symposium on Applications and the Internet* SAINT.

9

Yolum, P., and Singh M. P. (2003). Emergent Properties of Referral Systems. In *Proceedings of the 2nd Conference of AAMAS*, ACM Press, New York, 592-599.

10

Yu, B., and Singh, M. P. (2002). Emergence of agent-based referral networks. In *Proceedings of the 1st International Joint Conference of AAMS: Part 3,* ACM Press, New York, 1208-1209.

11

Yu, B., and Singh, M. P. (2003). Searching social networks. In *Proceedings of the 2nd International Joint Conference on AAMS '03*, ACM Press, New York, 65-72.

Biographies

Tadashi Nakano (**tnakano@ics.uci.edu**) is a postdoctoral researcher of the School of Information and Computer Science at University of California, Irvine. He received his M. E. and Ph.D. from Osaka University, Japan. His research interests include complex adaptive systems and evolutionary computation.

Tatsuya Suda (**suda@ics.uci.edu**) is a professor of the School of Information and Computer Science at University of California, Irvine. His current research focuses on application of biological principles and large complex system principles onto networks, high speed networks, next generation Internet, ATM (Asynchronous Transfer Mode) networks, object-oriented distributed systems, and multimedia applications.

Vishakh (**email@vishakh.com**) graduated with a Bachelor's degree inInformation & Computer Science with a specialization in artificialintelligence from the University of California, Irvine in 2004. He isthe co-founder of Medical World 4 U, a portal that seeks to brokerglobal medical services online. His research interests lie incomputational biology and artificial life.

Nicholas Urrea (**nurrea@uci.edu**) is an undergraduate student majoringin Information and Computer Science at the University of California,Irvine, where he is specializing in artificial intelligence. He isworking on the development of an interactive GPS-based game andconducting research on simian population dynamics.