A resource-allocation mechanism for multiagent networks

Vishakh, Nicholas Urrea, Tadashi Nakano, Tatsuya Suda
School of Information & Computer Science
University of California, Irvine
Irvine, CA 92697
mail@vishakh.com, nurrea@uci.edu, tnakano@ics.uci.edu, suda@ics.uci.edu

ABSTRACT
The nature of computer networks and the manner in which network services are provided are changing dramatically. Network architectures that employ virtual mobile agents to provide services are under development. We describe Price Propagation, which is based on pricing in market economies. It is a decentralized adaptive mechanism for regulating agent behavior, with the goal of allocating computational resources optimally.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *Distributed networks, Network communications, Wireless communication.*

C.2.1 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Intelligent agents, Multiagent systems.*

General Terms
Algorithms, Economics

Keywords
Multiagent systems, computer networking, artificial life

1. 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). However, there is now a proliferation of mobile wireless gadgets and other appliances on the Internet. As the number of network services and demand for them increase dramatically, traditional client- network services server methods look increasingly untenable.

Several initiatives are therefore underway to provide network services in a decentralized manner. Among these are multiagent systems such as [1] and [2], in which a number of virtual software entities, called agents, migrate over networks to interact with each other and provide network services. The chief advantages of these systems are:

- **Scalability:** They do not suffer from performance bottlenecks. In centralized systems, bottlenecks in computational power and bandwidth occur at the server's side as the number of clients increases. These bottlenecks can only be overcome through hardware and software upgrades, often at substantial prices. Multiagent 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 [3]. When demand fluctuates, agents replicate or delete themselves, thus adapting to the new conditions. Centralized 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. Furthermore, agents can provide services to users even when they are entirely cut off from the rest of the network.

In this paper, we describe an adaptive mechanism called Price Propagation, using which agent population sizes and distributions can be controlled in a distributed manner so that network resources are allocated optimally.

2. PRICE PROPGATION
2.1 The mechanism
In market economies, prices of commodities are used to gauge supply and demand patterns in an elegant and decentralized manner. For instance, a merchant dealing in Llama milk in Peru might look at milk prices in different cities and choose the one with the highest price. Prices can be used to spot trends in

Genetic and Evolutionary Computing Conference (GECCO) '05,
June 25–29, 2005, Washington, DC, USA.

overall supply and demand and also geographical trends such as, "There is a shortfall in milk in the south." Prices are used to solve the economic problem of where and when to allocate resources. When prices of a commodity reach equilibrium over the whole market, it indicates ideally that supply and demand are perfectly balanced and resources have been optimally allocated.

A similar pricing mechanism, called Price Propagation can be applied to multiagent networks. Instead of trading money for goods, users provide agents with "energy" in return for network services. Energy here is a virtual currency that users can purchase (or even get for free) from the owners of a network. Agents spot energy supply and demand trends over networks and react accordingly. Energy provided by users to agents is associated with a price. Each network node (e.g. a PC, PDA or mobile phone that a user is using) calculates this price and propagates it to its neighbors. Over several iterations, prices are dispersed over the network and agents are able to use them to effectively gauge supply and demand over the whole network and also over local regions. A marketplace is then created in which the actors are users (humans who need network services), network nodes (devices that provide computational resources to agents and users) and agents (software entities that use computational resources to provide services to users and other agents).Agents use pricing information in order to decide whether to replicate, migrate, die or stay put. In essence, this model is used to propagate information using prices.

2.2 Setting Prices

There are two kinds of prices- Local Price (LPs) and Global Price (GPs). While an LP is a local gauge of supply and demand specific to one node, a GP applies on a larger scale. LPs are calculated at each node:

LP = (Energy supply by users at node) / (No. of agents at node)

The GP is then calculated by each node using the LP and incoming GPs from adjoining nodes of the network. The GP, then, is the average of the incoming GPs and the LP:

GP = Average (Sum of incoming prices and Local Price)

Figure 1 shows a network of three nodes. We assume that each user provides a service-providing agent with 1 unit of energy per iteration. In our simulation, an iteration is a single execution of the main loop of our simulation program. In actual computer networks, it could mean a fixed time quantum within which each node is expected to propagate prices. On node N1, since there are two users and one agent, the LP is 2.0. Similarly, the LPs on N2 and N3 are 1.0 and 0.5 respectively. The GP on N1 is the average of the LP on N1 and the incoming GPs from the connected nodes, N2 and N3, i.e. 1.16. After prices propagate for a few iterations, they will converge to stable values which will provide indications of supply and demand.

2.3 Agent Behavior

The Price Propagation model uses price trends to regulate agent behavior. Variations in prices across networks nodes indicate imbalances in users' demand for and agents' supply of services. Agents spot these imbalances and act on them based on certain thresholds. These thresholds, in turn, adapt to price trends so that the system as a whole dynamically adapts according to changing network conditions.

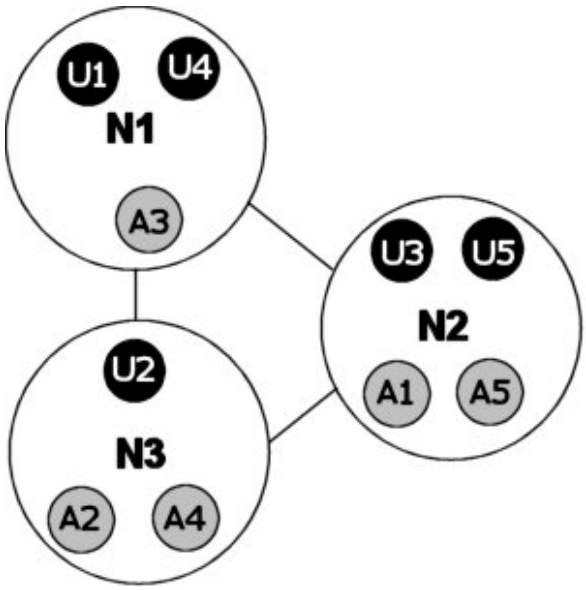

Figure 1- A Three-node Network

2.3.1.1 Migration

Migration by agents ensures that their distribution across networks conforms to service-seeking users' distribution. A higher GP on a neighboring node indicates to an agent that there is an excess of energy supply there (or in its general direction) and it would benefit both itself and the network if it were to migrate there. Thus, every agent compares the GP on its node to the highest incoming GP from directly-connected nodes. If the difference between the two is greater than its Migration Threshold, the agent migrates to the node from which the highest GP came.

2.3.2 Replication

Replication by agents ensures that their populations always grow in response to increased demand for services from users so that multiagent network services are scalable. Agents compare the GP on their node to a pre-defined ideal price. If the difference between the two is greater than an agent's Replication Threshold, then it replicates by creating a child agent. Half of the parent's accumulated energy is given to the child. In addition, the child's thresholds mutate randomly by a small amount to introduce greater variation in the system.

2.3.2.1 Death

Death ensures that agent populations always can scale down in response to drops in user demand and there is no wastage of computational resources. Every agent is charged a small platform cost in energy at every iteration the network node it resides on. This platform cost is predetermined and fixed across the network. If an agent does not receive significant amounts of energy consistently, then the platform cost ensures that its reservoir of energy is steadily depleted. Once an agent has no energy left, it is deleted.

2.3.2.2 Adaptation

Agent behavior in the Price Propagation model is heavily dependent on the Replication and Migration thresholds. However, these thresholds are set arbitrarily at the beginning and even if they reflect the initial conditions of networks, there is no guarantee that they will be at the correct levels when conditions change. In order to make these thresholds adaptive, they have to be constantly adjusted according to the prevailing prices. When the GP of an agent's node indicates that users demand more services than agents can provide, then the agent's thresholds are lowered slightly so that it is more likely to replicate and migrate to satiate users' need for services. Conversely, thresholds are raised slightly when there are more than enough agents needed to provide users with services. For the network in Figure 1, a GP higher than 1 indicates that thresholds should be lowered, while a GP less than 1 indicates that thresholds should be raised. The magnitude of the above threshold adjustments is a fixed value called the Adaptation Factor.

2.4 Price Equilibria

As mentioned before, the perfect balance of supply and demand is indicated when prices reach an equilibrium. The object of Price Propagation is to facilitate a dynamic balance between energy supply by users and energy demand by agents. Using prices and the associated agent behaviors, agent populations are distributed over networks to create such equilibria. One such case would be realized in the network in Figure 1 if agent A2 migrated to node N1, leading to optimal resource allocation. Equilibria such as these are the desired ends of the Price Propagation algorithm. Of course, perfect equilibria are not always possible in dynamic networks. Even so, even equilibrium-like conditions suggest a healthy supply between supply and demand.

3. SIMULATION

We now present the results of a computer simulation we performed to test Price Propagation. We simulated a 25-computer network with the initial Replication Threshold set to 0.4, Migration Threshold to 0.3 and Adaptation Factor to 0.001 for all agents. In the beginning, 25 agents were randomly scattered over the network. In any given iteration, there was a 1 in 100 chance at each node that a new user would appear and request the service. Once a user found a service-providing agent, it took 50 iterations before the user was satisfied. The simulation was run for 1,000 iterations and the trends were graphed.

The system as a whole adapted well to the shortfall in the initial agent population, given the number of users. This is borne out by looking at the mean Global Price for all nodes through the simulation in Figure 2. The mean GP steadily decreased as the agent population replicated and migrated, until it settled around the near-ideal value of 1.0.

Figure 3 shows the number of users present in the network though the simulation, while Figure 4 shows the agent population for the duration. The agent population can be seen increasing until it reached an adequate level and then stabilized. There were minor variations towards the end on account of the stabilization of threshold values.

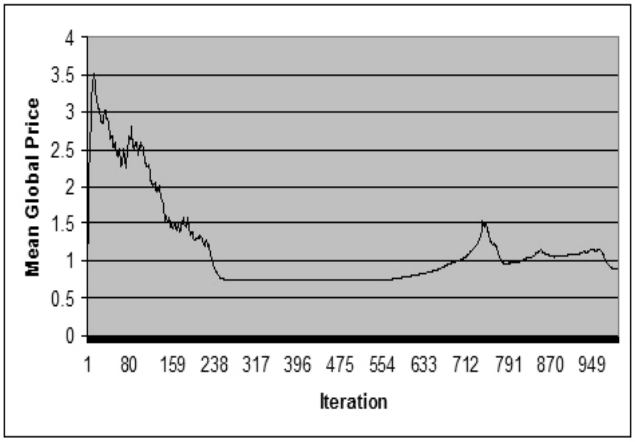

Figure 2- Global Price

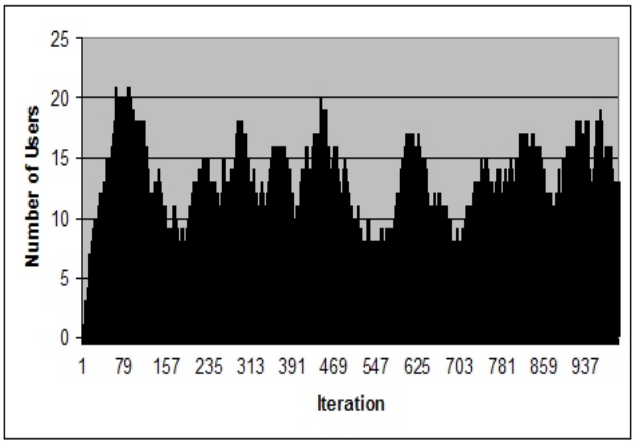

Figure 3- Number of Users

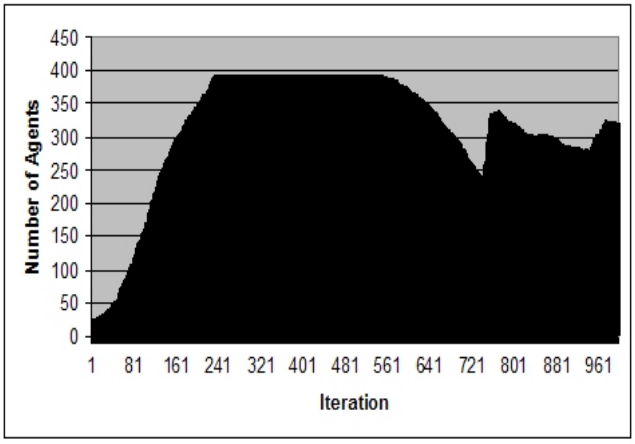

Figure 4- Agent Population

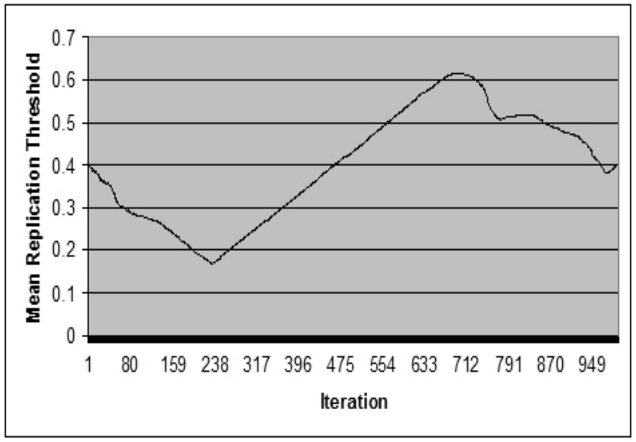

Figure 5- Replication Threshold

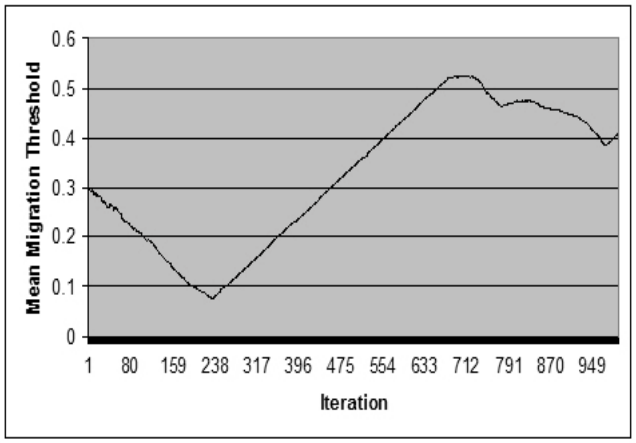

Figure 6- Migration Threshold

Figures 5 and 6 show how the Replication and Migration thresholds changed. The agents can be seen adapting *en masse*. Initially, thresholds kept dropping in value so that agents were more likely to replicate and migrate to respond to the shortfall of agents across the network. Once the agent population and distribution matched user demand, the thresholds increased steadily to reduce the heightened levels of replication and migration. Finally, both thresholds reached stable levels and showed minor variations according to changing user demand.

We can also see the utility of the death behavior in Figure 7. Initially, since there was a shortfall of agents and plenty of energy being supplied by users, there were no deaths. As the simulation progressed, the lowered Replication Threshold meant that there was eventually an excess of agents. Some agents began to be starved of energy and started dying. As a result, the overall agent population became sustainable and the number of deaths started dropping.

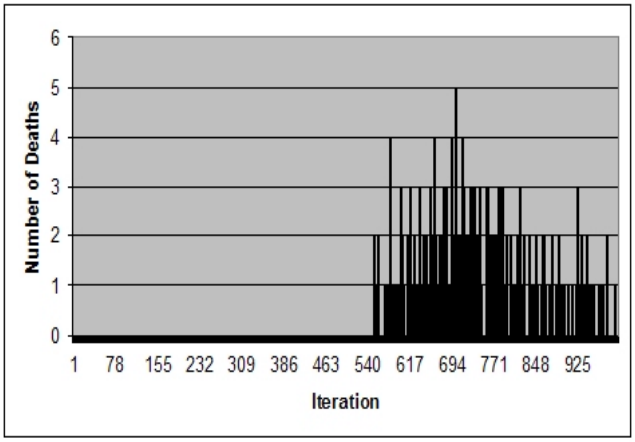

Figure 7- Deaths

4. CONCLUSION

Through simulations, we have confirmed that Price Propagation seems to be a promising mechanism for providing adaptive and evolvable multiagent network services. There is much scope for future work, including testing the model under more complex conditions and creating real-world applications based on it. The mechanism has parallels in other fields, such as quorum sensing in bacteria [4]. We plan to study these and draw insight from them. Finally, we see substantial potential for applying similar mechanisms to other domains where dynamic, decentralized resource-allocation is needed.

5. REFERENCES

[1] Wang, M., and Suda, T. 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*, 2001.

[2] Carzaniga, A., Picco, G.P., and Vigna, G. Designing distributing applications with mobile code paradigms. In *Proceedings of the 19th international conference on Software Engineering* (May 17-23, Bostom, Ma.) ACM Press 1997, pp. 22-32.

[3] Lerman, K., and Galstyan, A. Agent memory and adaptation in multi-agent systems. In *Proceedings of the second internaional joint conference on AAMS 03'* (July 14-18, Melbourne, Australia) ACM Press 2003, pp. 797-803.

[4] Miller, M.B. and Bassler, B.L. Quorum Sensing in Bacteria. In *Annual Review of Microbiology.* Vol. 55 (2001), 165-199.