MAKING GRID SYSTEMS SELF-ORGANIZING AND ADAPTIVE: AN APPROACH LEVERAGING BIOLOGICAL CONCEPTS AND MECHANISMS

Paskorn Champrasert and Junichi Suzuki
Department of Computer Science
University of Massachusetts Boston,
Boston, MA 02125-3393
{paskorn, jxs}@cs.umb.edu

ABSTRACT
Grid computing systems are expected to be more scalable and adaptive. Based on the observation that various biological systems have already overcome these requirements, the proposed architecture, called SymbioticSphere, applies biological concepts and mechanisms to design grid systems (application services and middleware platforms). In SymbioticSphere, each application service and middleware platform is designed as a biological entity, and implements biological concepts and mechanisms such as decentralization, energy exchange, migration, replication and death. Simulation results show SymbioticSphere exhibits self-organization with inherent support of scalability and adaptability through collective actions and interactions of application services and middleware platforms. Preliminary simulation results show that application services and middleware platforms collectively adapt to dynamic changes in the network (e.g. user location, network traffic and resource availability).

KEY WORDS
Autonomous adaptive agents, Autonomic grid system, biologically-inspired computing

1. Introduction

Grid systems are expected to autonomously scale to enormous demand placed upon them, survive from partial systems failures and adapt to dynamic network environments in order to improve user experience, expand system's operational longevity and reduce maintenance cost [1, 2, 3]. Based on the observation that various biological systems have already achieved these requirements (i.e. autonomy, scalability, survivability and adaptability), the proposed network architecture, called SymbioticSphere, applies biological concepts and mechanisms to design network systems (application services and middleware platforms) [1]. The authors of the paper believe if grid systems adopt certain biological concepts and mechanisms, they may be able to meet these requirements.

In SymbioticSphere, each application service and middleware platform is designed as a biological entity, analogous to an individual bee in a bee colony. An application service is implemented as an autonomous and distributed software agent. Each agent implements a functional service and follows simple behaviors similar to biological entities, such as replication, death, migration and energy exchange. A middleware platform runs on a network host and operates agents. Each platform implements a set of runtime services that agents use to perform their services and behaviors, and follows biological behaviors such as replication, death and environment sensing.

Similar to biological entities, agents and platforms in SymbioticSphere store and expend *energy* for living. Each agent gains energy in exchange for performing its service to other agents or human users, and expends energy to use network and computing resources. Each platform gains energy in exchange for providing resources to agents, and continuously evaporates energy to the network environment. SymbioticSphere models agents and platforms as different species, and follows several concepts in ecological food chain to determine how much energy agents/platforms expend at a time and how often they expend energy. The abundance and scarcity of stored energy affect behaviors of an agent/platform. For example, an abundance of stored energy is indicates higher demand for the agent/platform; thus the agent/platform may be designed to favor replication in response to higher energy level. A scarcity of stored energy (an indication of lack of demand) may cause death of the agent/platform.

Similar to biological systems, SymbioticSphere exhibits self-organizing emergence of desirable system characteristics such as scalability and adaptability. These characteristics emerge from collective behaviors and interactions of agents and platforms, rather than they are not present in any single agent/platform. Simulation results show that agents and platforms autonomously scale to rapid demand changes and adapt to dynamic changes in the network (e.g. user location and resource availability). In certain circumstances, agents and platforms spontaneously cooperate in a symbiotic manner to pursue their mutual benefits (i.e. to increase their scalability and adaptability), although each of them is not designed to do so.

[1] SymbioticSphere is an extension to the Bio-Networking Architecture [4, 5, 6, 7]. The Bio-Networking Architecture was adopted by Object Management Group as a part of its standard specification [8].

This paper is organized as follows. Section 2 summarizes key design principles of SymbioticSphere. Section 3 describes the designs of agents and platforms. Section 4 shows simulation results Sections 5 and 6 conclude with discussion on related work and future work.

2. Design Principles in SymbioticSphere

SymbioticSphere consists of two major system components: agents (applications services) and middleware platforms. Agents run on platforms, which in turn run on network hosts. Agents and platforms are designed based on the three principles described below.

- **Decentralization:** Agents and platforms are decentralized. There are no central entities that control or coordinate agents/platforms (i.e. no directory servers and no resource managers). Decentralization allows agents/platforms to be scalable, survivable and simple by avoiding a single point of performance bottleneck and failure [9, 10] and by avoiding any central coordination in deploying agents/platforms [11].
- **Autonomy:** Agents and platforms are autonomous. They monitor their local network environments, and based on the monitored environmental conditions, they autonomously behave and interact with each other without any interventions from/to other agents, platforms and human users.
- **Adaptability:** Agents and platforms are adaptive to changing environment conditions (e.g. user demands, user locations and resource availability). Adaptation is achieved through designing agent/platform behavior policies to consider local environment conditions. For example, agents may implement a migration policy of moving towards a platform that forwards a large number of request messages for their services. This results in the adaptation of agent locations, and agents concentrate around the users who request their services. Also, platforms may invoke replication and death behaviors when their energy levels become over and below thresholds. This results in the adaptation of platform population, and platforms adjust resource availability on them against the demands for resources.

3. SymbioticSphere

This section describes the designs of agents and platforms.

3.1. Agents

Each agent consists of three parts: *attributes*, *body*, and *behaviors*. *Attributes* carry descriptive information regarding the agent, such as agent ID, energy level, description of a service it provides, and price (in energy units) of the service. *Body* implements a service that the agent provides. For example, an agent may implement a genetic algorithm for an optimization problem, while another one may implement a mathematical model for scientific simulations. *Behaviors* implement actions that are inherent to all agents. Although SymbioticSphere defines nine standard agent behaviors [5, 12], this paper focuses on three of them

- *Migration:* Agents may migrate from one platform to another.
- *Replication*: Agents may make copies of themselves as a result of abundance of energy. A replicated (child) agent is placed on the platform that its parent agent resides on, and it receives the half amount of the parent's energy level.
- *Death*: Agents may die due to energy starvation. When an agent dies, an underlying platform removes the agent from the network and releases all resources allocated to the agent.

3.2 Platforms

Each platform runs on a network host and operates agents[2]. It consists of *attributes*, *behaviors* and *runtime services*. *Attributes* carry descriptive information regarding the platform, such as platform ID, energy level and healthy level. *Healthy level* is defined as a function of the age of and resource availability on a network host that the platform runs on. The age indicates how long a network host remains alive (i.e. how much stable a network host is). Resource availability indicates how much resources (e.g. memory space) are available for agents and platforms on a network host. Healthy level affects behaviors of a platform and agent. For example, higher healthy level indicates higher stability of and higher resource availability on a network host that the platform resides on. Thus, the platform may be designed to replicate itself on a healthier neighboring host than the current local host. This results in the adaptation of platform locations. Platforms work on stable and resource rich network hosts.

Behaviors are the actions that are inherent to all platforms. Although SymbioticSphere defines six standard platform behaviors [12], this paper focuses on two of them.

- *Replication.* Platforms may make copies of themselves as a result of abundance of energy (i.e. high demand for resources available on them). The child platform receives the half amount of the parent's energy level.
- *Death.* Platforms may die due to the lack of energy. A dying platform uninstalls itself from the network and releases all resources the platform uses. Despite the death of a platform, an underlying network host remains active so that other platforms can run on it in the future.

Runtime services are services that agents and platforms use to perform their behaviors. In order to maximize de-

[2] Currently, SymbioticSphere assumes that at most one platform runs on each network host.

centralization and autonomy, they only use their local runtime services. They are not allowed to invoke any runtime services running on a remote platform.

3.3 Behavior Policies of Agents and Platforms

Each agent and platform has policies for its behaviors. A behavior policy defines when to and how to invoke a particular behavior. Each behavior policy consists of one or more *factors* (F_i), which evaluate environment conditions (e.g. resource availability on a local platform) or agent/platform status (e.g. energy level and healthy level). Each factor is given a certain *weight* (W_i) relative to its importance. Behaviors are invoked if the weighted sum of factor values ($\Sigma F_i * W_i$) exceeds a threshold.

Agent migration behavior has three factors listed below. If there are multiple neighboring platforms that an agent can migrate to, the agent calculates a weighted sum of the above factor values for each of the platform, and migrates to a platform that generates the highest weighted sum.

- *Service Request Ratio*, (# of service requests on a remote platform)/(# of service requests on a local platform), which encourages agents to move towards users.
- *Healthy Level* ratio, (healthy level on remote platform - healthy level local on local platform) / (healthy level on local platform), which encourages agents to move towards platforms running on healthier hosts.
- *Migration interval*, interval from the time of a previous migration, which discourages too many agents to migrate too often.

Agent replication and death behaviors have a factor that evaluates the current energy level of agent.

Platform replication behavior has a factor described below. A replicated (child) platform is placed on a host whose healthy level is highest among neighboring hosts[2].

- *Healthy Level Ratio*, (healthy level on a remote host)/(healthy level on a local host), which encourages platforms to replicate themselves on a healthier host.

Platform death behavior has a factor that evaluates the current energy level of platform. Each platform never performs death behavior while an agent(s) runs on the platform.

Each agent/platform incurs energy loss (i.e. behavior cost) to invoke behaviors except death behavior. When the energy level of an agent/platform goes over the cost of a behavior, the agent/platform decides if it performs the behavior by calculating a weighted sum of factor values.

3.4 Energy Exchange among Agents and Platforms

SymbioticSphere models agents and platforms as different biological species, and follows several ecological concepts to determine how often agents/platforms expend energy and how much energy they expend at a time. Fig. 1 shows a simplified energy flow in the ecological system. The sun gives light energy, and producers (e.g. plants and microorganisms) convert it to chemical energy. The chemical energy flows through multiple species, called consumers. It will be eventually transferred to decomposers (e.g. bacteria and fungi). For example, shrubs (producers) convert the sun light energy to chemical energy, hares (primary consumers) consume shrubs, and foxes (secondary consumers) consume hares. When energy is transferred from one species to another, it is known that about 10% of the energy maintained by one species goes to another species [13]. The remaining 90% is used for metabolism, growth and actions/behaviors (e.g. moving).

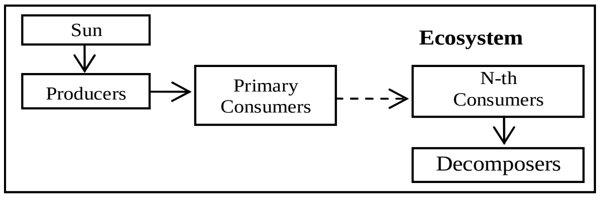


Fig. 1 Energy Flow in Ecosystem

Fig. 2 shows the energy exchange in SymbioticSphere. SymbioticSphere models users as the sun, agents as producers, and platforms as (primary) consumers. Similar to the sun, users have unlimited amount of energy. Agents gain energy from users[3], and pay energy to consume resources provided by platforms. They pay 10% of the current energy level to platforms. Platforms gain energy from agents, and pay (evaporate) 10 % of the current energy level to the environment.

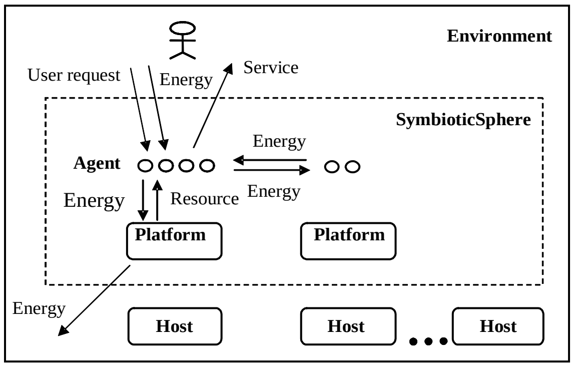
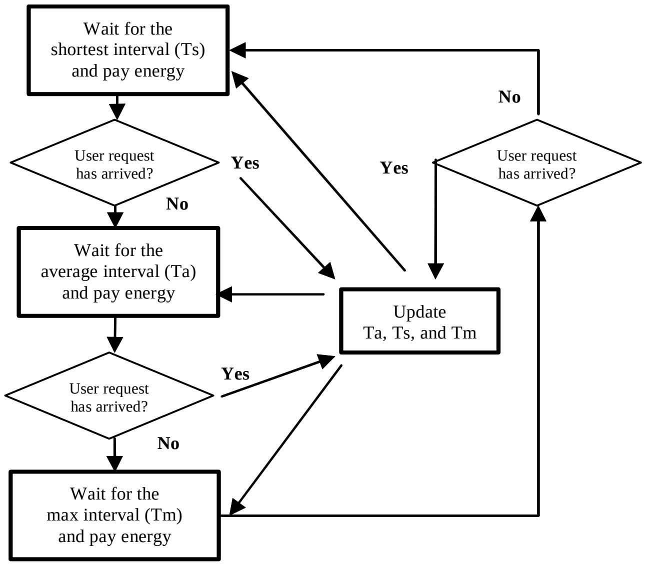


Fig. 2 SymbioticSphere

Agents dynamically change the rate of transferring energy to platforms, based on the rate of accepting service requests from users. When agents process more service requests, they consume more resources. Thus, agents transfer energy (i.e. 10% of the current energy level) to platforms more often. In contrast, they reduce their energy transfer rate in response to lower energy intake from users.

[3] Each agent specifies, in its body, the price (in energy units) of service that it provides.

In order to dynamically change energy transfer rate, each agent keeps an interval time between an incoming service request and a previous request. It records the average, shortest and maximum intervals of previous *N* service requests (*Ts*, *Ta* and *Tm*, respectively). Fig. 3 shows how often each agent transfers energy to platforms. First, an agent waits for *Ts* and pay energy to an underlying platform. Then, the agent checks if a new service request(s) has arrived during the previous *Ts* interval. If arrived, the agent updates *Ts*, *Ta* and *Tm* values, waits for *Ta*, and then pays energy to a platform. Otherwise, it waits for *Ta* and pays energy to a platform. Similarly, each agent repeats energy transfers in *Ts*, *Ta* and *Tm* intervals.



Wait for the shortest interval (Ts) and pay energy
User request has arrived?
Yes
No
Wait for the average interval (Ta) and pay energy
User request has arrived?
Yes
No
Wait for the max interval (Tm) and pay energy
Update Ta, Ts, and Tm
User request has arrived?
Yes
No

Fig. 3 Energy Transfer between Agents and Platforms

Ta is a simple moving average calculated from the intervals of previous *N* service requests. The shortest and longest intervals play a role of weighted values to make energy transfer rate follow dynamic changes in service request rate. *Ts* and *Tm* values are periodically reset (every *M* service requests). A previous simulation result shows that the proposed energy exchange mechanism allows agents to change their energy expenditure rate well against dynamic change in energy intake [12].

Platforms also dynamically change the rate to evaporate energy, depending on the rate of accepting energy transfers from agents. The more often they receive energy transfer from agents, the more often they evaporate energy (10% of the current energy level). Each platform changes its energy evaporation rate in the same way as each agent changes its energy expenditure rate. (i.e., each platform follows the mechanism described in Fig. 3.)

4. Preliminary Simulation Results

This section shows preliminary simulation results to evaluate how the biologically-inspired mechanisms in SymbioticSphere impact on scalability and adaptability of grid systems (i.e. agents and platforms)[4].

In this paper, adaptability is evaluated as *service adaptation* and *resource adaptation*. *Service adaptation* is the activities to adaptively improve the quality and availability of services provided by agents. Quality of service is measured as response time of agents for service requests from users. Service availability is measured as the number of available agents. *Resource adaptation* is the activities to adaptively improve availability of resources provided by platforms and efficiency to utilize the resources. Resource availability is measured as the number of platforms that make resources available for agents. Resource efficiency indicates how many service requests can be processed against resource utilization.

A simulated network is a 8x8 grid topology network with 64 network hosts (Fig. 4). At the beginning of each simulation, a platform is initialized on the network host 63, and an agent is deployed on the platform. Transmission latency is 0.1 second between two network hosts. (one

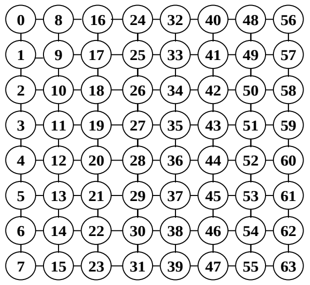


Fig. 4 Simulated Network

simulation cycle corresponds to 1 sec in simulation time.) Each network host has 256MB memory[5]. Out of the memory space, an operating system consumes 128 MB and Java virtual machine consumes 64MB. Thus, 64MB is available for a platform and agents on each network host. Each agent and platform consumes 5 MB and 20 MB, respectively. This assumption is obtained from a prior empirical experiment [5].

4.1 Single User Simulations

In this simulation study, a user is placed on the network host 19, and propagates service requests for 24 hours (from 0:00 to 24:00) at a rate shown in Fig. 5. Three simulation scenarios are implemented. Scenario 1 initializes a platform as non-biological entity (on the network host 63); thus it does not replicate and die. There is only one platform running throughout a simulation. Scenario 2 initializes a non-biological platform on each network host

[4] Simulations were carried out with the SymbioticSphere simulator, which contains 14,100 lines of Java code. This simulator is freely available at http;//dssg.cs.umb.edu/symbiosis/ for researchers who investigate autonomic grid systems.

[5] Currently, memory availability represents resource availability.

(i.e. 64 platforms on 64 network hosts). The platforms do not replicate and die. In Scenario 3, a biological platform is initialized on network host 63. The platform can dynamically replicate and die based on the behavior policy described in Section 3. In either scenario, agents are implemented as biological entities; they can migrate, replicate and die. A key simulation objective is to investigate the behavior of agents and platforms in Scenario 3 against the other two extreme cases (i.e. Scenarios 1 and 2).

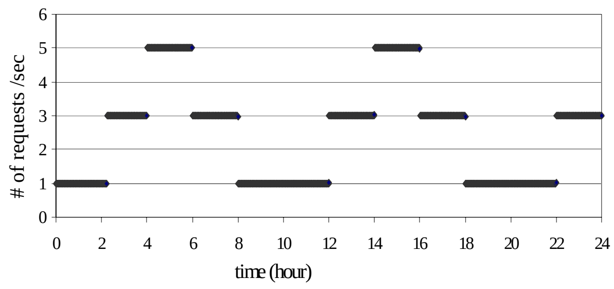


Fig. 5 Change in Service Request

Fig. 6 shows how service availability (i.e. the number of agents) changes. In Scenarios 1, 2 and 3, agents autonomously adapt their population to demand change for their services. When service request rate becomes high, agents gain more energy and replicate themselves. In contrast, when service request rate becomes low, some agents die due to energy starvation since they cannot balance energy gain and expenditure. Biological mechanisms contribute for agents to improve service availability as a group.

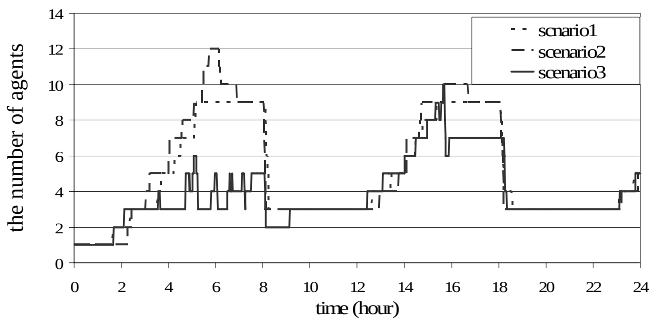


Fig. 6 the Number of Agents

Fig. 7 shows how resource availability (i.e. the number of platforms) changes. Since platforms are not designed as biological entities in Scenarios 1 and 2, the number of platforms do not change (i.e. one and 64 platforms in Scenarios 1 and 2, respectively). In Scenario 3, in which platforms are designed as biological entities, platforms dynamically adapt their population to demand change for resources. When service request rate becomes high, agents gain more energy and transfer more energy to platforms. As a result, platforms replicate themselves more often. In contrast, when service request rate becomes low, some platforms die due to energy starvation since they cannot gain enough energy to keep their population. Fig. 7 shows biological mechanisms contribute for a group of platforms to adaptively improve resource availability.

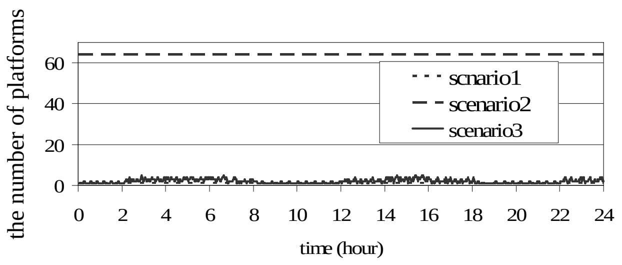


Fig. 7 the Number of Platforms

Fig. 8 shows the average distance between a user and agents (in network hop counts). In Scenario 1, the distance remains constant (5 hop counts) because a platform does not make its children platforms that agents may migrate to. In Scenario 2, the distance rapidly decreases because agents can migrate to platforms that are closer to a user. The distance becomes zero in 1.5 hours. (agents reach user's location in 1.5 hours.) In Scenario 3, the distance gradually decreases and becomes zero in 8 hours. Unlike Scenario 2, Scenario 3 begins with a single (biological) platform. At the beginning of a simulation, the platform needs to wait for agents to grow their population and transfer it enough energy so that it can replicate itself on a neighboring host. If the child platform is placed on a host that is closer to a user, agents move to the platform, thereby decreasing the distance to a user by one hop count. This process takes more time than how agents move toward a user in Scenario 2.

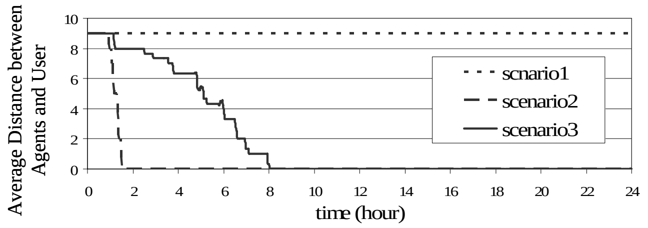


Fig. 8 Average Distance between Agents and User

Fig. 9 depicts the average distance between a user and platforms. It shows platforms gradually move toward a user, although platform replication policy does not consider user location. This is an example of symbiotic emergence. If replicated platforms are placed on hosts that agents want to migrate to (i.e. hosts closer to users), the platforms will survive. Otherwise, they will die because agents do not migrate to them and transfer energy to them. In a sense, agents indirectly instruct platforms where to replicate themselves. This results in a mutual benefit for agents and platforms. Agents can work closer to users and gain more energy from the users, and platforms gain more energy from agents.

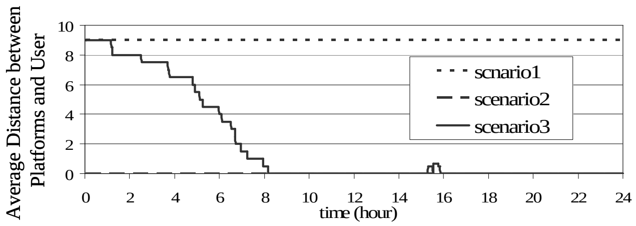


Fig. 9 Average Distance between Platforms and User

Fig. 10 shows the quality of services (i.e. the average response time for agents to process service requests from a user). In the first three hours, response time becomes very high in either scenario, because agents have to store energy for a while to start replications. After the first three hours, agents increase their population to process more service requests (Fig 10); thereby decreasing response time dramatically. In Scenario 1, response time is greater than in Scenarios 2 and 3 because agents do not migrate toward a user. Please note that response time include transmission latency between two network hosts. (i.e. the closer agents work from a user, the shorter their response time becomes). In scenario 2, which is the best case scenario for the response time measurement, agents migrate toward a user, and response time drops to 1 second in four hours. In scenario 3, agents have to wait for platforms to accumulate enough energy to start replications. Response time drops to 0.5 second in eight hours. Fig. 14 shows biological mechanisms contribute for agents and platforms to collectively adapt response time for users.

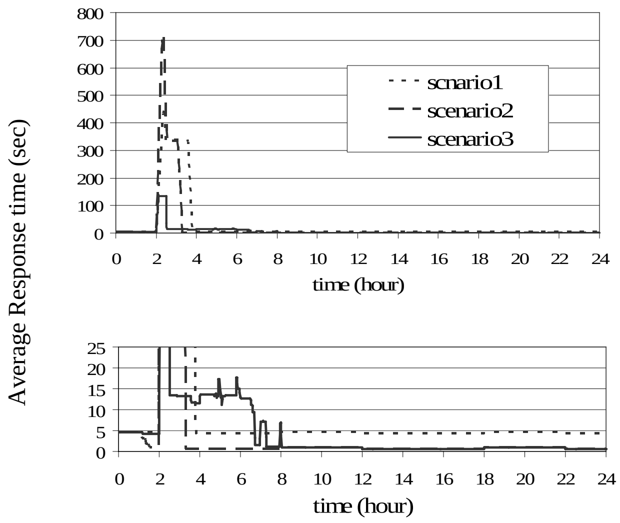


Fig. 10 Average Response Time

Fig. 11 shows resource efficiency. It is measured as (the total number of user requests processed by agents) / (the total amount of resources consumed by agents and platforms). Scenario 1 is the best case scenario because only one platform is used to process all service requests. Scenario 2 is the worst case scenario because all service requests are processed by 64 platforms including idle ones that do not operate agents. In Scenario 3, both agents and platforms dynamically change their populations. Resource efficiency in Scenario 3 is often close to the best case result in Scenario 1. Fig. 11 shows that biological mechanisms contribute for agents and platforms to autonomously keep resource efficiency high.

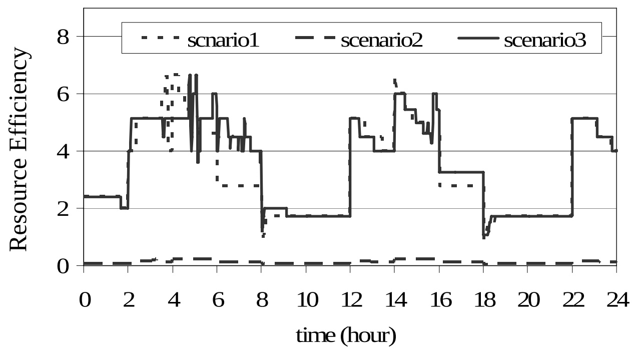


Fig. 11 Resource Efficiency

4.2 Multiple Users Simulations

A series of the following simulations is to evaluate how agents and biological platforms behave in a network environment where multiple users propagate service requests.

Fig. 12 shows how each user changes its service request rate. A user always resides on each of the hosts 9 and 14, and the third user enters the network (on the host 59).

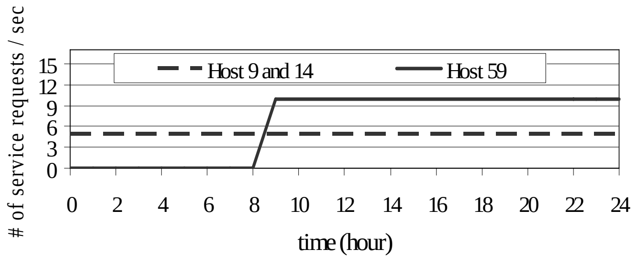


Fig. 12 Change in Service Request

Fig. 13 shows how service availability (i.e. the number of agents) changes. When service request rate spikes at 8:00, agents autonomously increase their population rapidly because they gain more energy from users and perform replication more often. This result shows agents scale well to rapid changes in service demand.

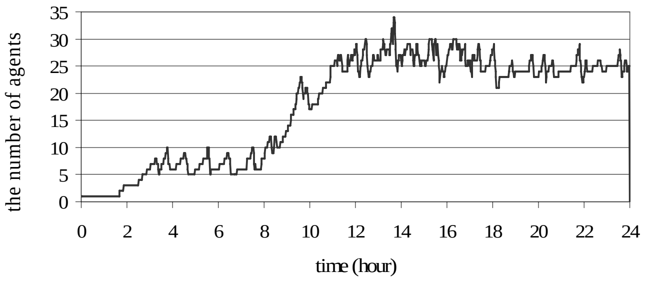


Fig. 13 the Number of Agents

Fig. 14 shows how resource availability (i.e. the number of platforms) changes. Platforms adaptively adjust resource availability by changing their population against the demand for their resources.

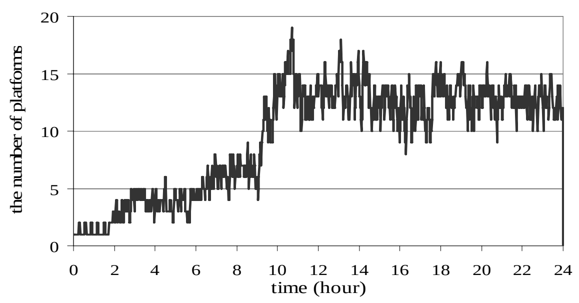


Fig. 14 the Number of Platforms

Fig. 15 shows quality of service (i.e. the average response time for agents to process service requests from users). In the first two hours, response time becomes very high because agents have to store energy for a while to start replications. After that, response time dynamically drops to 20 seconds at 2:00. Then, agents start migrating towards users. Response time decreases to 10 seconds at 8:00. When the third user enters the network, response time increases to 20 seconds. Two hour later, it decreases to 10 seconds again because agents work close from the third user. Fig. 15 shows that agents autonomously adapt their response time against dynamic changes in service request rate and the number of users by adjusting their population and locations.

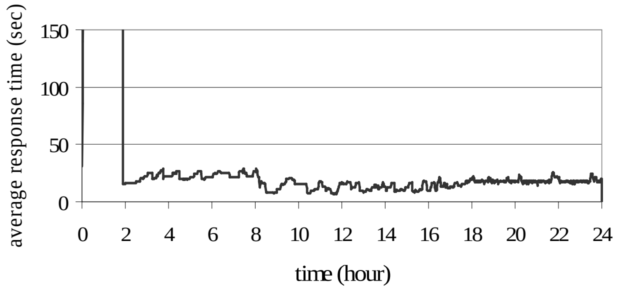


Fig. 15 Average Response Time

Fig 16 shows how workload (i.e. service requests) is distributed over available platforms. Load Balancing Index (LBI) is measured with the Equation 1 (LBI is a standard deviation of x_i).

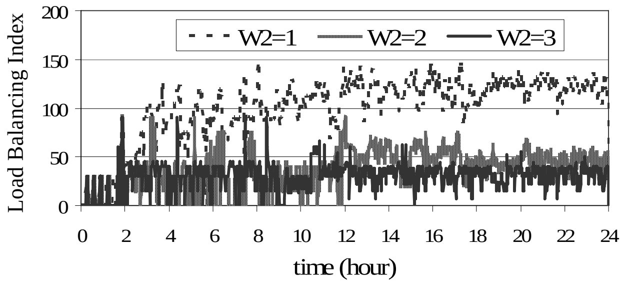


Fig. 22 Load Balancing Index

$$Load \quad Balancing \quad Index = \sqrt{\frac{\sum_i (X_i - \mu)^2}{N}} \qquad (1)$$

x_i indicates (the number of messages processed by agents running on platform i) / (resource utilization on platform i). μ represents the expected average of x, which is means (the total number of messages processed by all agents) / (the total amount of resource utilization on all platforms) / (the number of platforms; N). W2 in Fig. 16 indicates the weight value for healthy level ratio factor in agent migration policy. When a higher weight value is given, agents distribute workload more aggressively. Application designers can configure this value based on the requirements to their applications. Fig. 16 also shows that agents and platforms gradually increase the degree of load balancing over time. This is an example of symbiotic emergence. Agent migration behavior policy encourages agents to move towards platforms running on healthier hosts. Platform replication behavior policy encourages platforms to replicate themselves on healthier hosts. As a result, service requests are processed by agents that are spread over the platforms running on healthy hosts. This contributes to balance workload per platform, although agent migration policy and platform replication policy do not consider agent population, platform population nor load balancing. This results in a mutual benefit for both agents and platforms. Platforms help agents decrease response time by making more resources available for them. Agents help platforms to keep their stability by distributing workload on them (i.e. by avoiding excessive resource utilization on them).

5. Related Work

This work is an extended work to the Bio-Networking Architecture, as described in Section I. In the Bio-Networking Architecture, agents are designed as biological entities, and they achieve service adaptation in a decentralized and collective manner [4, 6]. However, platforms are static and non-biological entities. Since they do not change their population and locations dynamically, they cannot achieve resource adaptation. In SymbioticSphere, agents and platforms achieve both service adaptation and resource adaptation in a decentralized, collective and symbiotic manner.

Resource Broker [14] proposes a resource adaptation mechanism for grid systems. In this mechanism, a centralized system component monitors heterogeneous environments where different network hosts have different levels of stability and different resource availability. Given monitored environment conditions, the mechanism adapts resource allocations for grid applications. Unlike Resource Broker, SymbioticSphere service adaptation as well as resource adaptation with decentralized agents and platforms.

[15] and [16] propose generic adaptation frameworks for grid systems. They can be used to achieve both service adaptation and resource adaptation. In these frameworks, centralized system components store the current environment conditions, and decide which adaptation strategy to execute against the monitored conditions. In contrast,

SymbioticSphere does not assume any centralized system components. Each of agents and platforms collects and stores environment conditions, and autonomously decide which behavior to invoke.

The concept of energy in SymbioticSphere is similar to money in economy. MarketNet [17] applies the concept of money to achieve market-based access control for network applications. However, it does not mention the details on how much and how often application components make payments with each other. SymbioticSphere currently focuses on service adaptability and resource adaptability. It also provides application developers the details on energy exchange between system components (i.e. agents and platforms) so that they can consistently develop adaptive network systems.

6. Conclusions

This paper overviews the architectural design of SymbioticSphere, and presents how it implements biological concepts and mechanisms to make grid systems (i.e. services and platforms) scalable, survivable and adaptive. This paper also describes how agents and platforms interact with each other to collectively exhibit emergence of desirable system characteristics (e.g. adaptability). Preliminary simulation results show that agents and platforms collectively adapt to dynamic changes in the network (e.g. user location, network traffic and resource availability) in a decentralized and autonomous manner.

An extended set of simulations is planned to investigate how the proposed biologically-inspired mechanisms impact on scalability, survivability and adaptability of grid systems. For example, future simulations will operate agents and platforms on larger heterogeneous networks where different network hosts have different resource availability, intermittent unstable networks where network hosts and network links between them can be occasionally down, and mobile networks where users dynamically move.

Reference:

[1] P. Dini, W. Gentzsch, M. Potts, A. Clemm, M. Yousif and A. Polze, “Internet, Grid, Self-adaptability and Beyond: Are We Ready?,” In *Proc. of the IEEE International Workshop on Self-Adaptable and Autonomic Computing Systems*, August 2004.

[2] R. Sterritt and D. Bustard, “Towards an Autonomic Computing Environment,” In *Proc. of 14th IEEE International Workshop on Database and Expert Systems Applications*, September 2003.

[3] Large Scale Networking Coordinating Group of the Interagency Working Group for Information Technology Research and Development (IWG/IT R&D), Report of Workshop on New Visions for Large-scale Networks: Research and Applications, March 2001.

[4] T. Suda, T. Itao and M. Matsuo, “The Bio-Networking Architecture: The Biologically Inspired Approach to the Design of Scalable, Adaptive, and Survivable/Available Network Applications,” In *K. Park (ed.) The Internet as a Large-Scale Complex System*, Oxford University Press, June 2005.

[5] J. Suzuki and T. Suda, “A Middleware Platform for a Biologically-inspired Network Architecture Supporting Autonomous and Adaptive Applications” In *IEEE Journal on Selected Areas in Communications (JSAC)*, vol. 23, no. 2, February 2005.

[6] J. Suzuki, “Biologically-inspired Adaptation of Autonomic Network Applications,” In *International Journal of Parallel, Emerging and Distributed Computing*, vol. 20, no. 2, June 2005.

[7] T. Nakano and T. Suda, "Adaptive and Evolvable Network Services," *In Proc. of the Genetic and Evolutionary Computation Conference*, 2004.

[8] S. Sameshima, J. Suzuki, S. Steglich and T. Suda, *Platform Independent Model (PIM) and Platform Specific Model (PSM) for Super Distributed Objects*, Object Management Group, Final Recommended Specification, 95 pages, November 2004.

[9] N. Minar, K. H. Kramer and P. Maes, “Cooperating Mobile Agents for Dynamic Network Routing,” In *A. L. G. Hayzelden and J. Bigham (eds.) Software Agents for Future Communications Systems*, Springer, 1999.

[10] R. Albert, H. Jeong and A. Barabasi, “Error and Attack Tolerance of Complex Networks,” *Nature* 406, 2000.

[11] G Cabri, L. Leonardi and F Zambonelli, “Mobile-Agent Coordination Models for Internet Applications,” *IEEE Computer*, February 2000.

[12] P. Champrasert and J. Suzuki, "SymbioticSphere: A Biologically-inspired Network Architecture for Autonomic Grid Computing," In *Proc. of the 2nd IEEE/Create-Net Int’l Workshop on Networks for Grid Applications*, Boston, MA, October 2005. to appear.

[13] R. M. Alexander, “ Energy for Animal Life,” Oxford University Press, May 1999.

[14] A. Othman, P. Dew, K. Djemame, I, Gourlay, “Adaptive Grid Resource Brokering,” In IEEE International Conference on Cluster Computing, December 2003

[15] S. Cheng, D. Garlan, B. Schmerl, P. Steenkiste, and N. Hu, “Software Architecture-based Adaptation for Grid Computing,” In *the 11th IEEE Conference on High Performance Distributed Computing*, July 2002.

[16] K Shirose, S Matsuoka, H Nakada, and H Ogawa, “Autonomous Configuration of Grid Monitoring Systems,” In *the 2004 Symposium on Application and the Internet*, January 2004.

[17] M. P. Wellman, “A Market-Oriented Programming Environment and Its Application to Distributed Multicommodity Flow Problems,” Journal of Artificial Intelligence Research, Vol. 1, 1993