SymbioticSphere: A Biologically-inspired Network Architecture for Autonomic Grid Computing

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

***Abstract*—Grid computing systems are expected to be more scalable, more survivable from partial systems failures and more adaptive to dynamic network environments in order to improve user experience, expand system's operational longevity and reduce maintenance cost. Based on the observation that many biological systems have already overcome these requirements, the proposed network 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 an artificial biological entity, analogous to an individual bee in a bee colony. Application services and middleware platforms implement biological concepts and mechanisms such as decentralization, energy level, healthy level, energy exchange between species, environment sensing, migration, replication and death. Like in biological systems, desirable system characteristics such as scalability, survivability and adaptability emerge from the collective actions and interactions of application services and middleware platforms. This paper presents the architectural design of SymbioticSphere, and describes how application services and middleware platforms act and interact with each other. 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).**

I. INTRODUCTION

Autonomic 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]. In order to realize such autonomic grid systems, the authors of the paper observe that various biological systems have already developed the mechanisms necessary to achieve key requirements of autonomic grid systems such as autonomy, scalability, survivability and adaptability. The authors of the paper believe if grid systems adopt certain biological concepts and mechanisms, they may be able to meet these requirements.

The SymbioticSphere architecture applies key biological concepts and mechanisms to design grid systems (application services and middleware platforms)[1]. In SymbioticSphere, each application service and platform is modeled as an artificial 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, pheromone emission and environment sensing. A middleware platform runs on a network host and operates agents (i.e. application services). Each middleware 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 entities in the biological world, 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 environment. SymbioticSphere models agents and platforms as different biological 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 various behaviors of an agent/platform. For example, an abundance of stored energy is an indication of higher demand for the agent/platform; thus the agent/platform may be designed to favor replication in response to higher level of stored energy. A scarcity of stored energy (an indication of lack of demand) may eventually cause death of the agent/platform.

Similar to biological systems, SymbioticSphere exhibits emergence of desirable system characteristics such as scalability, survivability and adaptability. These characteristics emerge from the collective behaviors and interactions of agents and platforms, rather than they are not present in any single agent/platform. Agents and platforms act autonomously, influenced by local environment conditions (e.g. network traffic and resource availability) and local interactions with other agents and platforms.

This paper overviews the architectural design of SymbioticSphere, and presents how agents and platforms implement biological concepts and mechanisms such as decentralization, energy level, healthy level, energy exchange, environment sensing, migration, replication and death. This paper also

[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 for Super Distributed Objects [8].

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 autonomously adapt to dynamic changes in the network (e.g. user location, network traffic 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 adaptability), although each of them is not designed to do so.

This paper is organized as follows. Section II summarizes key design principles of SymbioticSphere. Section III overviews the architecture of SymbioticSphere and describes biological mechanisms implemented in agents and platforms. Section IV shows preliminary simulation results to evaluate the impact of the proposed biologically-inspired mechanisms on adaptability of grid systems. Sections V and VI conclude with discussion on related work and future work.

II. Design Principles in SymbioticSphere

SymbioticSphere consists of two major system components: agents (applications services) and middleware platforms (Fig. 1). Agents run (or live) on platforms, which in turn run on network hosts. Agents and platforms are designed based on the three principles described below, in order to collectively make grid systems scalable, survivable and adaptive.

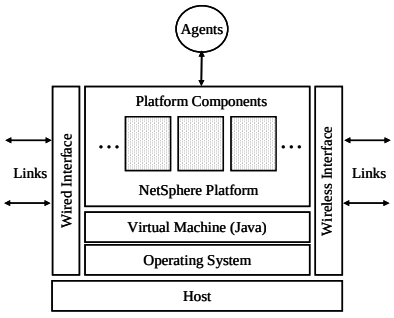

Fig. 1 SymbioticSphere Components

(1) Decentralization: Agents and platforms in SymbioticSphere are decentralized. There are no central entities that collect environment conditions in the network, and controls or coordinate agents/platforms (i.e. no directory servers and no resource managers). Agents/platforms act based on local environment conditions and local interactions with other agents/platforms. Decentralization can improve system's scalability and survivability by avoiding a single point of performance bottleneck and failure [9, 10] and by avoiding any central coordination in deploying agents/platforms [11].

(2) Autonomy: Agents and platforms in SymbioticSphere 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 intervention from/to other agents, platforms and human users.

(3) Adaptability: Agents and platforms in SymbioticSphere 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 instance, agents may invoke migration behavior of moving towards neighboring platforms that forward a large number of service requests from users. This results in the adaptation of agent locations; 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.

III. SymbioticSphere

This section describes how agents and platforms implement a series of biological concepts and mechanisms, and how they interact with each other.

A. Agents

Each agent consists of three parts: *attributes*, *body* and *behaviors* (Fig. 2). *Attributes* carry descriptive information regarding the agent, such as agent ID[2], age, energy level, owner's name, description of a service the agent provides, and price (in energy units) of the service that the agent provides. Each agent has a remotely accessible interface that allows human users and other agents to read its attribute values [5].

Body implements a service that the agent provides and contains materials relevant to the service. For instance, an agent may implement a genetic algorithm for a large-scale optimization problem, while another agent may implement a physical and mathematical model for scientific simulations. An agent that implements a genetic algorithm for an optimization problem may contain artificial genes, instructions for mutation and crossover, and fitness functions.

Behaviors implement non-service related actions that are inherent to all agents. They control autonomous actions of each agent, as described in Section II. Agents can have an arbitrary number of behaviors. Some example agent behaviors are explained below.

- *Migration:* Agents may migrate from one platform to another.
- *Communication*. Agents may communicate with other agents for the purposes of, for instance, requesting a service or exchanging energy.

[2] Agents and platforms maintain globally unique IDs. See [5] for more details on the design of globally unique IDs.

- *Energy exchange and storage*: Agents may receive and store energy in exchange for providing services to other agents. Agents also expend energy. For instance, agents may pay energy units for services that they receive from other agents. In addition, when an agent uses resources on a platform (e.g. CPU and memory), it may pay energy units to the platform.

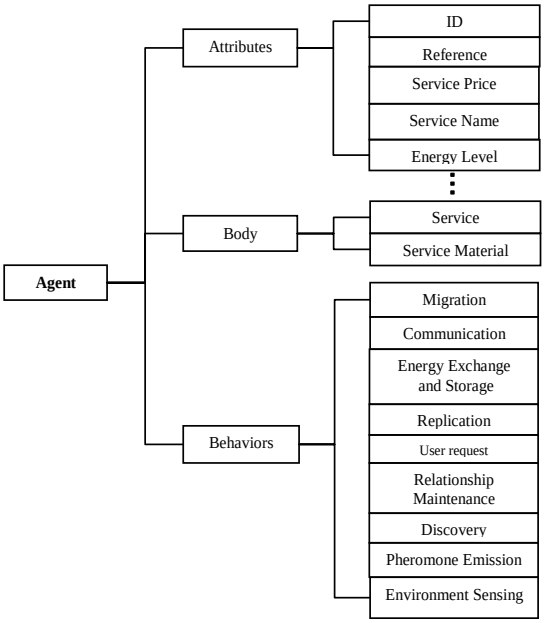

Fig. 2 Agent Design in SymbioticSphere

- *Replication*: Agents may make a copy of themselves as a result of abundance of energy.
- *Death*: Agents may die as a result of lack of energy. If energy expenditure of an agent is not balanced with the energy units it receives from providing services to other agents, it will not be able to pay for the resources it needs, i.e., it dies from lack of energy. When an agent is dead, an underlying platform removes the agent from the network environment and releases all the resources allocated to the agent.
- *Relationship maintenance*: Agents may establish and maintain a limited number of relationships with other agents. A relationship contains information regarding the partner agent, for instance, the attributes of the partner agent. Relationships are autonomously maintained by the participant agents. Such relationships may have a variety of uses, including performing discovery to search for agents.
- *Discovery:* Agents may seek for other agents of certain attributes by forwarding queries to agents that they have relationships to.
- *Pheromone emission:* Agents may emit and leave a pheromone (or a trace) behind on a platform when they migrate to another platform. This is to indicate their presence to other agents. A pheromone contains the emitter's ID and a reference to the platform that the emitter migrated to. Pheromones are emitted with certain strength and may decay over time. Pheromones may have a variety of uses, including improving the performance of discovery.
- *Environment sensing.* Agents may sense their local environment. For instance, an agent may sense which agents are in local and neighboring platforms and what services they provide. An agent may also sense pheromones (e.g. which agents left pheromones on local and neighboring platforms) and resources (e.g. CPU cycles and memory space available on local and neighboring platforms).

B. Platforms

Platforms are execution environments (or middleware) for agents. Each platform runs on a network host and operates agents[3]. It abstracts low-level operating and networking details (e.g. network I/O and concurrency control for executing agents), and aids developing and deploying agents in the network [5].

Each platform consists of three parts; *attributes*, *behaviors* and *runtime services* (Fig. 3). *Attributes* carry descriptive information regarding the platform, such as platform ID[2], age, energy level and healthy level. Each platform has a remotely accessible interface that allows human users, agents and other platforms to read its attribute values [5].

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 local host remains alive in the network. It represents how much stable the local host is. Resource availability indicates how much resources (e.g. CPU cycles, memory space, disk space and network bandwidth) are available for a platform and agents on the host.

Healthy level affects various behaviors of a platform and agent. For example, higher healthy level indicates higher stability of and/or 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 concentrate around stable and resource-rich network hosts. Also, lower healthy level indicates that the platform runs on a network host that is unstable and/or poor in resources. If agents are designed to migrate towards healthier (i.e. more stable and resource-rich) network hosts, the platform will eventually die due to energy starvation because agents do not come and pay energy. This results in the adaptation of platform population. Platforms and agents avoid running on the network hosts that are unstable and/or poor in resources.

[3] Currently, SymbioticSphere assumes that there is only one platform running on a network host.

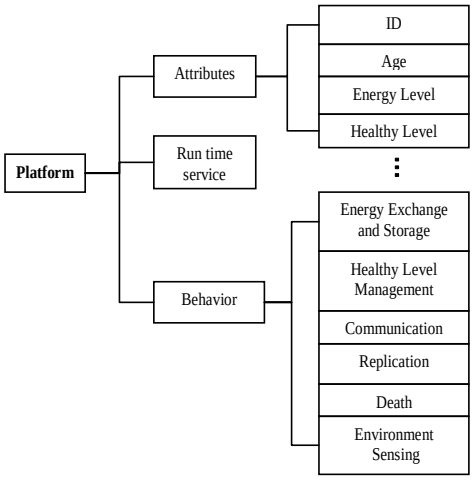

Fig. 3 Platform design in SymbioticSphere

Behaviors are the actions that are inherent to all platforms. They control autonomous actions of each platform (see Section II). Platforms can have an arbitrary number of behaviors. Some example platform behaviors are explained below.

- *Energy exchange and storage*: Platforms may receive and store energy in exchange for providing resources to agents. Platforms may also periodically evaporate energy to the network environment. Evaporated energy is no longer available for any agents and platforms.
- *Healthy level management*: Platforms may monitor and maintain healthy level of the network hosts that they reside on. Platforms may also inquire of neighboring platforms for the healthy level of their underlying network hosts.
- *Communication*. Platforms may communicate with other platforms for the purposes of, for instance, inquiring the current healthy level.
- *Replication.* Platforms may make a copy of themselves as a result of abundance of energy (i.e. an indication of higher demand for the platforms). A replicated (child) platform may be placed on a healthier host than the host where its parent platform resides on[3].
- *Death.* Platforms may die due to 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 in the future.
- *Environment sensing*: Platforms may sense their local environment. For instance, a platform may sense how many and which agents are running on local and neighboring platforms. A platform may also sense resources (e.g. CPU cycles, memory space, disk space and network bandwidth) available on local and neighboring platforms.

Runtime services are middleware services that agents and platforms use to invoke their behaviors. For example, each platform provides the lifecycle management service, which implements agent replication and death behaviors. Agents use the runtime service to perform replication or death behaviors [5]. In order to maximize the degree of decentralization and autonomy of agents/platforms, they only use the local runtime services on the platform they reside. They are not allowed to invoke any runtime services running on remote platforms. Please also note that platforms do not provide global environment information, such as a list of agents/platforms running on the entire network or the locations of agents/platforms. Agents and platforms know their local environment information, such as a list of agents running on local and neighboring platforms, or resources available on local and neighboring platforms.

C. 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 functions, or *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 (ΣF_i*W_i) exceeds a threshold. For example, the factors in the agent migration behavior may include:

- *Distance to users*, which encourages agents to move towards users requesting the services provided by the agent.
- *Mutual repulsion*, which encourages agents to repel with each other.
- *Healthy level*, which encourages agents to migrate to a network host whose healthy level is higher.

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 whether it performs the behavior by calculating a weighted sum of factor values.

D. Energy Exchange between Agents and Platforms

As described earlier, agents and platforms are modeled as biological entities. In the biological world, living entities try to maximize their energy gain, while minimizing their energy expenditure, in order to live longer and produce more offspring. Similarly, in SymbioticSphere, agents and platforms strive to gain more energy to live longer and produce more offspring. For example, agents may move towards users (i.e. energy sources) so that they can gain more energy from the users. Platforms may replicate themselves on healthier network hosts so that the replicated (child) platforms can attract agents and gain energy from them.

SymbioticSphere models agents and platforms as different types of biological entities (i.e. different species), and follows several concepts in ecological food chain. Fig. 4 shows a sim-

simplified energy flow in the ecological system. The sun gives light energy, and producers (e.g. plants and microorganisms) convert the light energy to chemical energy. The chemical energy flows through multiple types of entities (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.

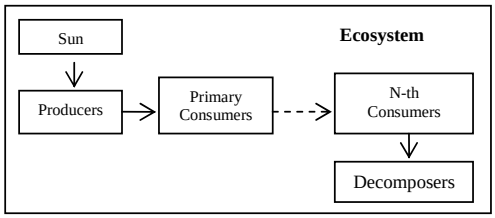

Fig. 4 Energy Flow in Ecosystem

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 [12]. The remaining 90% is used for metabolism, growth and actions/behaviors (e.g. moving).

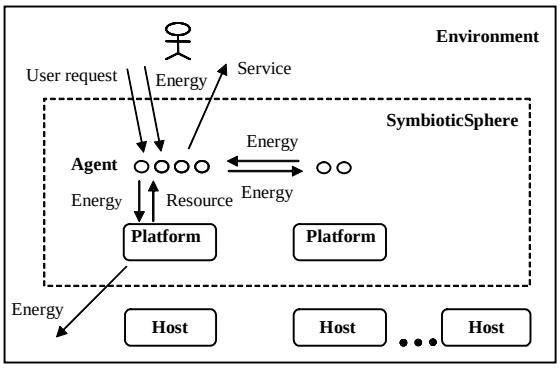

Fig. 5 SymbioticSphere

Fig. 5 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. They pay energy units for services provided by agents. Agents gain energy from users[4], 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.

Agents dynamically change the rate of transferring energy to platforms, depending on the rate of incoming service requests from users. When agents process more service requests from users, they consume more resources. Therefore, agents transfer energy units (i.e. 10% of the current energy level) to platforms more often. On contrary, they reduce their energy transfer rate in response to lower energy intake from users.

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. 6 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.

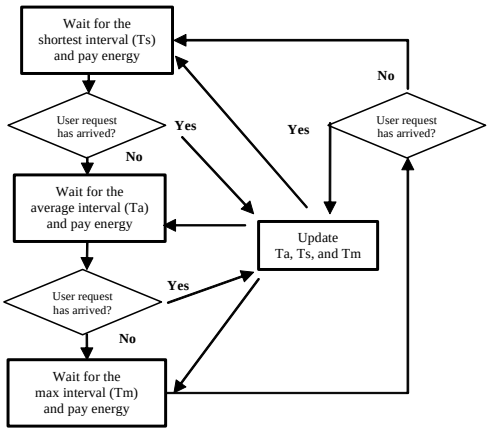

Fig. 6 Energy Transfer Scheme in Agents and Platforms

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).

Platforms dynamically change the rate to evaporate energy, depending on the rate of incoming 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. 6.)

IV. Preriminary Simulation Results

This section evaluates the biologically-inspired mechanisms in SymbioticSphere through simulation. This paper shows preliminary simulation results to examine how they impact on adaptability of grid systems. Simulations were carried out with the SymbioticSphere simulator, which contains

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

13,500 lines of Java code. It can run arbitrary number of agents, platforms, users and network hosts on simulated networks. This simulator is freely available at http://dssg.cs.umb.edu/symbiosis/ for researchers who investigate autonomic grid systems (Fig. 7).

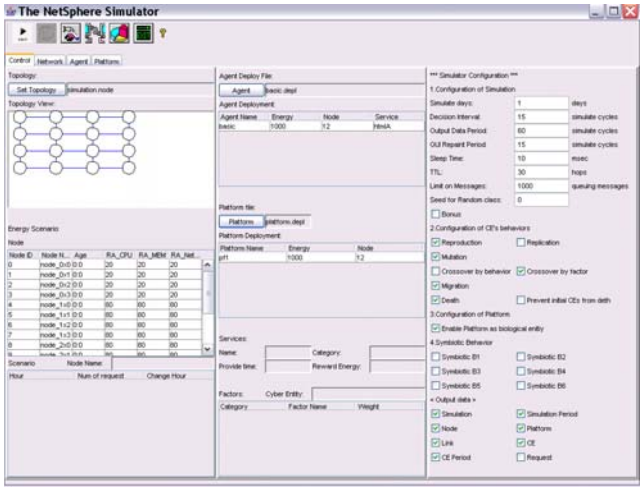

Fig 7 Screenshot of the SymbioticSphere Simulator

In this paper, adaptability is defined as *service adaptation* and *resource adaptation*. *Service adaptation* is the activities to adaptively improve the quality and availability of services provided by agents. The quality of services is measured as response time of agents for service requests from users. Service availability is measured as the number of agents. *Resource adaptation* is the activities to adaptively improve availability and efficiency for utilizing resources provided by platforms. Resource availability is measured as the number of platforms that makes resources available for agents. Resource efficiency is measured as (the total number of user requests processed by agents) / (the total amount resources consumed by agents and platforms).

A. Results of Energy Exchange Simulations

In order to evaluate whether the energy exchange scheme described in Section III.D works well, an agent is deployed on a platform to accept service requests from a user. The agent receives service requests and energy units from a user, and pays some of the energy units to an underlying platform.

Fig. 8 shows how service request rate changes during 24 hours in simulation time. Fig. 9 shows how much energy the agent expends. The agent gains 10 energy units from a user in exchange for providing its service. In this simulation, two types of agents are implemented for evaluation purpose. The first type of agent implements the energy exchange scheme described in Section III.D. It uses "average," "shortest," and "longest" intervals in its energy expenditure cycle (see Section III.D). The second type of agent uses the only "average" interval in its energy expenditure cycle.

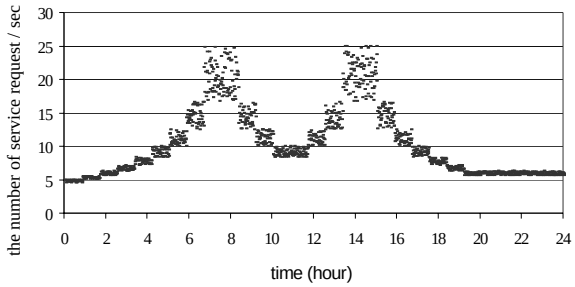

Fig. 8 Change in service request

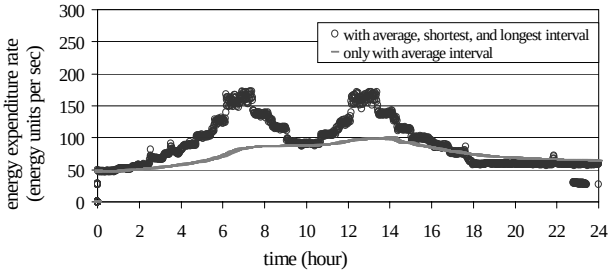

Fig. 9 Energy Expenditure Rate

Fig. 9 shows that the proposed energy exchange scheme allows agents to change their energy expenditure rate against dynamic change in energy intake. Agent energy expenditure follows energy intake (or service request rate) well. In the energy exchange scheme only using "average" interval, agent energy expenditure does not follow energy intake (or service request rate) well.

B. Configurations of Adaptability Simulations

A series of the following simulations evaluate how agents and platforms achieve service adaptation and resource adaptation in a collective and symbiotic manner.

A simulated network is a 4x4 grid topology network with 16 network hosts (Fig. 10). At the beginning of each simulation, a platform is initialized on network host 12, and an agent is deployed on the platform. A user is placed on network host 7. Transmission latency is 0.1 second between two network hosts (one simulation cycle corresponds to 1 second in simulation time). Resource availability on each host/platform is quantified in *resource units*. The more resource units a host has, the more resources are available on the host. Each agent and platform consumes 9 and 30 energy units, respectively.

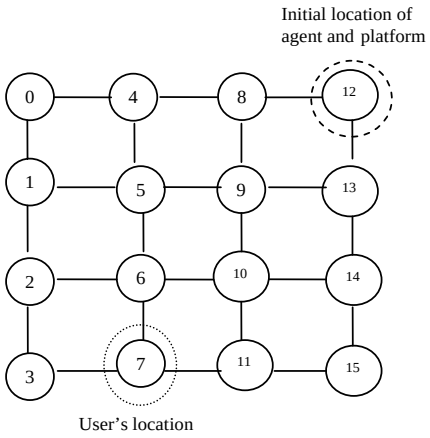

Fig. 10 Simulated Network

A user keeps propagating service requests according to a configured rate. Fig. 11 shows how service request rate changes for 24 hours (from 0:00 to 24:00).

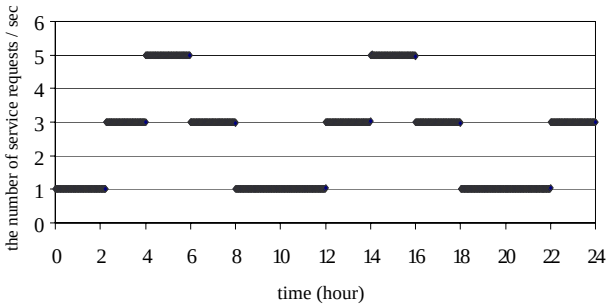

Fig. 11 Change in Service Request

Agents implement replication, migration and death behaviors. Migration behavior policy considers three factors described below

- *Service requests*, (# of service requests on a remote platform)/(# of service requests on a local platform), which encourages agents to move towards a user.
- *Agent population*, (# of agents on a local platform)/(# of agents on a remote platform), which encourages agents to move toward a less-crowded platform.
- *Migration interval*, the interval from a previous migration, which discourages too many agents to migrate too often.

Weight values for these factors are one. If the current energy level of an agent exceeds 2000, the agent calculates a weighted sum of factor values (see Section III.C). Then, it invokes the migration behavior if the weighted sum exceeds a threshold (2.0). If there are multiple neighboring platforms, the agent calculates a weighted sum of factor values for each of the neighboring platforms, and migrates to a platform that generates the highest weighted sum. The behavior cost of migration is 100 energy units.

Replication behavior is invoked when agent energy level exceeds 3000. The behavior cost of replication is 100 energy units. 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 behavior is invoked when agent energy level becomes zero.

Each agent processes a service request in 0.2 second, and gains 10 energy units from a user in exchange for each request processed. The initial energy units of each agent is 1000.

Platforms implement replication and death behaviors. Replication behavior policy considers the following factor.

- *Healthy level*, (# of service requests on a remote platform)/(# of service requests on a local platform), which encourages platforms to perform replication.

Weight values for this factor is one. If the current energy level of a platform exceeds 3000, the platform calculates a weighted sum of factor value (see Section III.C). It invokes the replication behavior if the sum exceeds a threshold. If there are multiple network hosts where a parent platform can replicate itself on, the platform calculates a weighted sum for each of the neighboring network hosts, and choose the network host that generates the highest healthy level. A replicated (child) platform receives the half amount of the parent's energy level. The behavior cost of replication is 100 energy units.

Death behavior is invoked when platform energy level becomes zero.

Pseudo code of running users, agents and platforms in simulations is shown in Figure 12.

```
initialize platforms, agents and users
While (not simulation last cycle)
    For each user do
        send service requests to one of the nearest agents
        according to a configured service request rate.
    End For
    For each platform do
        make a decision on replication and death behaviors.
        update healthy level.
        expend (evaporate) energy.
    End For
    For each agent do
        If (a service request(s) received) do
            process the request(s) and gain energy.
        End If
        make a decision on replication, migration and death behaviors
        update average response time and distance to users
        expend energy to a local platform.
    End For
End While
```

Fig. 12 Pseudo Code of Simulation Algorithm

C. Results of Adaptability Simulations

Three simulation scenarios are implemented to evaluate the adaptability of agents and platforms. Scenario 1 initializes a platform as non-biological entity (on network host 12); 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 (i.e. 16 platforms on 16 network hosts). The 16 platforms do not replicate and die. In Scenario 3, a biological platform is initialized on network host 12. The platform can replicate and die based on the behavior policy described in a previous section. In either scenario, agents are implemented as biological entities; they can migrate, replicate and die. A key simulation objective is to investigate agents and platforms in Scenario 3 against two extreme cases (i.e. Scenarios 1 and 2).

Fig. 13 shows how service availability (i.e. the number of agents) changes against dynamic service request rate (Fig. 8). In Scenarios 1, 2 and 3, agents adapt their population to changes in service request rate. 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.

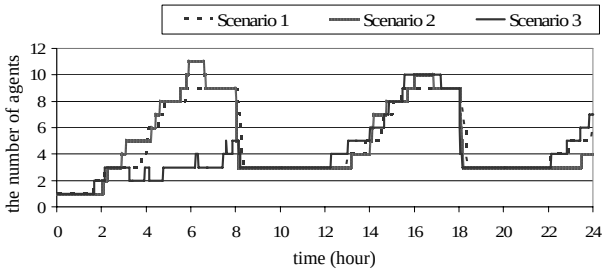

Fig. 13 The Number of Agents

Fig 14 shows how resource availability (i.e. the number of platforms) changes against dynamic service request rate (Fig. 8). Since platforms are not designed as biological entities in Scenarios 1 and 2, the number of platforms do not change (i.e. one and 16 platforms in Scenarios 1 and 2, respectively). In Scenario 3, in which platforms are designed as biological entities, the number of platforms dynamically changes against service request rate. When service request rate becomes high, agents gain more energy and transfer more energy to platforms. Then, in response to abundance of stored energy, platforms replicate themselves. In contrast, when service request rate becomes low, some platforms die due to energy starvation since they cannot gain enough energy from agents to keep their population. Fig. 14 shows that biological mechanisms contribute for a group of platforms to adaptively improve resource availability as a group.

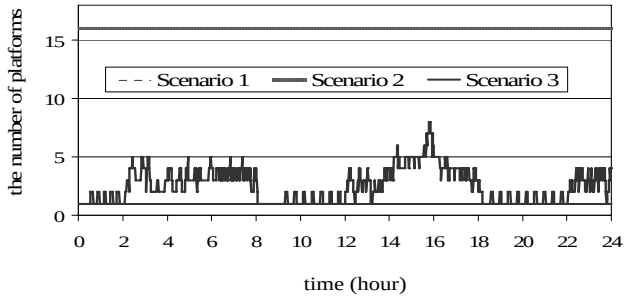

Fig. 14 The Number of Platforms

Fig. 15 shows the average distance between agents and user in network hop counts. In Scenario 1, the distance remains constant (5 hop counts) because a platform cannot make its children platforms that agents may migrate to. In Scenario 2, in which a platform is statically initialized on every network host, the distance between agents and user rapidly decreases because agents can migrate to platforms that are closer to a user. The distance becomes zero in 1.5 hours (i.e. 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 network host. If the child platform is placed on a network 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.

Another important observation from Fig. 15 is that 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 a user), the platforms will survive. Otherwise, they will die because agents do not migrate onto them and transfer energy to them. In a sense, agents indirectly instruct platforms where to replicate themselves. This results in a mutual benefit for both agents and platforms. Agents can work closer to a user and gain more energy, and platforms gain more energy.

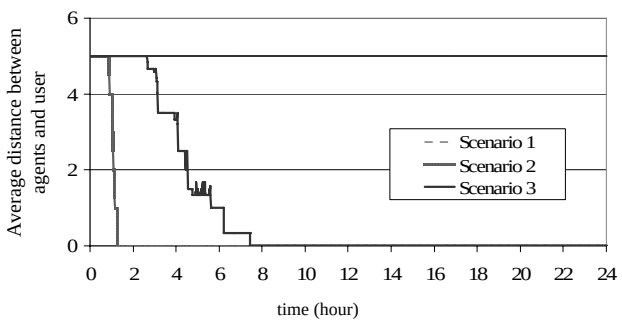

Fig. 15 Average Distance between Agents and User

Fig. 16 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 replication. After the first three hours, agents increase their population to process more service requests (Fig 11); 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 to 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 0.5 second in three hours. In scenario 3, agents have to wait for platforms to accumulate enough energy to start replication. Response time drops to 0.5 second in four hours. Fig. 16 shows biological mechanisms contribute for agents and platforms to collectively adapt agent response time.

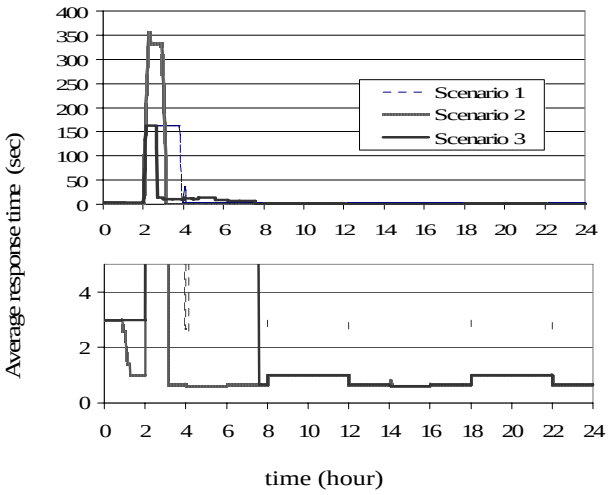

Fig. 16 Average Response Time

Fig. 17 shows resource efficiency, which indicates how many service requests can be processed per resource unit. It is measured as (the total number of user requests processed by agents) / (the total amount resources consumed by agents and platforms). Scenario 1 is the best case scenario in this case 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 16 platforms including idle ones that do not operate agents. The result is constantly very low. In Scenario 3, both agents and platforms adapt their population to dynamic service request rate. Resource efficiency in Scenario 3 is often close to the best case result in Scenario 1. Biological mechanisms contribute for agents and platforms to keep resource efficiency high.

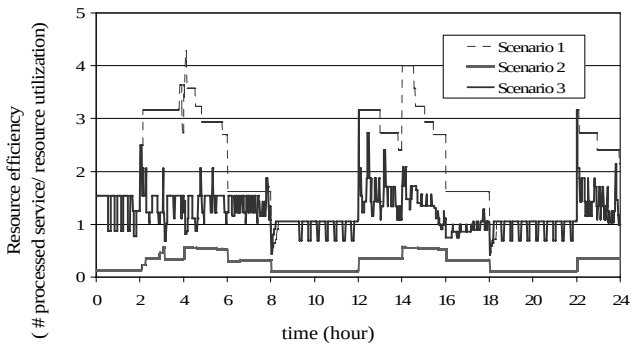

Fig. 17 Resource Efficiency

V. 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, 5, 6, 7]. 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 [13] 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.

[14] and [15] 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 [16] 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.

VI. Concluding Remarks

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.

References

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

[13] A. Othman, P. Dew, K. Djemame, I, Gourlay, “ Adaptive Grid Resource Brokering”, In IEEE International Conference on Cluster Computing (CLUSTER’03), 172, Dec 2003

[14] 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 (HPDC’02)*, July 2002.

[15] K Shirose, S Matsuoka, H Nakada, and H Ogawa, “Autonomous Configuration of Grid Monitoring Systems,” In *the 2004 Symposium on Application and the Internet (SAINT2004)*, Japan, January 2004.

[16] M. P. Wellman, “ A Market-Oriented Programming Environment and Its Application to Distributed Multicommodity Flow Problems,” Journal of Artificial Intelligence Research, Vol. 1, pp. 1-22,1993