Building Self-Configuring Data Centers with Cross Layer Coevolution

Paskorn Champrasert and Junichi Suzuki
Department of Computer Science, University of Massachusetts, Boston
Email: {paskorn, jxs}@cs.umb.edu

Abstract—**This paper describes a biologically-inspired architecture, called SymbioticSphere, which allows data centers to autonomously adapt to dynamic environmental changes. SymbioticSphere follows biological principles such as decentralization, evolution and symbiosis to design application services and middleware platforms in a data center. Each service and platform is designed as a biological entity, and implements biological behaviors such as energy exchange, migration, reproduction and death. Each service/platform also possesses behavior policies, as genes, each of which defines when to and how to invoke a particular behavior. This paper presents a set of behaviors for services and platforms, and describes how services and platforms act and interact with each other. Simulation results show that services and platforms autonomously adapt to dynamic network conditions (e.g., user location, network traffic and resource availability) by evolving their behavior policies across generations. Simulation results also show that services and platforms coevolve to improve their adaptability by adjusting their behavior policies cooperatively.**

Index Terms—**Autonomic self-configuring network systems, Biologically-inspired networking, evolvable network systems**

I. Introduction

Data centers are integral components to operate large-scale network applications. As they are rapidly increasing in complexity and scale, they face several challenges, particularly *autonomy* and *adaptability*. Data centers are expected to autonomously adapt to dynamic conditions in the network (e.g., network traffic and resource availability) in order to improve user experience, expand operational longevity and reduce maintenance cost [1, 2].

In order to meet these challenges (i.e., autonomy and adaptability), the authors of the paper propose to apply key biological principles and mechanisms to design data centers. This is motivated by an observation that various biological systems have already developed the mechanisms necessary to achieve autonomy and adaptability. For example, bees act autonomously, influenced by local environmental conditions and local interactions with other bees. A bee colony adapts to dynamic environmental conditions. When the amount of honey in a hive is low, many bees leave the hive to gather nectar from nearby flowers. When the hive is nearly full of honey, most bees remain in the hive and rest.

SymbioticSphere is an architecture that applies biological principles and mechanisms to design data centers. It consists of two kinds of components: *application services* and *middleware platforms*. Each of them is modeled as a biological entity, analogous to an individual bee in a bee colony. They are designed to follow several biological principles such as decentralization, emergence, evolution and symbiosis. An application service is designed as a software agent. Each agent implements a functional service (e.g., web service) and biological behaviors such as energy exchange, reproduction, migration and death. A middleware platform runs on a network host and operates agents. Each platform provides runtime services that agents use to perform their services and behaviors, and implements biological behaviors such as energy exchange, reproduction and death. SymbioticSphere models agents and platforms as different biological species.

In SymbioticSphere, each agent and platform autonomously senses its surrounding environment conditions and adaptively invokes a behavior suitable for the conditions. For example, an agent may invoke the migration behavior to move toward a network host that receives a large number of user requests for its services. This results in the adaptation of agent location; the agent can improve its response time to user requests. Also, a platform may invoke the reproduction behavior to make its offspring on a neighboring network host where resource availability is high. This results in the adaptation of resource availability; the platforms provide more resources to agents.

In addition to these (regular) behaviors, agents and platforms implement a special type of behaviors: *symbiotic behaviors*. A symbiotic behavior is a sequence of regular behaviors that an agent and its underlying platform invoke in order. As described above, agents and platforms can adapt to dynamic network environments by performing regular behaviors; however, regular behaviors of one species (e.g., agents) can degrade the adaptation of the other species (e.g., platforms) in some circumstances. For example, if too many agents migrate toward a user, the platforms near from the user have a risk to crash due to overloading or resource extinction. Symbiotic behaviors are intended for agents and platforms to balance and

Based on “Exploring Adaptive Data Centers through Cooperative Symbiotic Networking”, by P. Champrasert and J. Suzuki, which appeared in the Proceedings of the IEEE International Workshop on Future Trends of Distributed Computing Systems 2007, Sedona, AZ, USA, March 2007. © 2007 IEEE.

augment their adaptability by allowing the two species to cooperate for pursuing their mutual benefits.

Each agent/platform possesses *behavior policies*, each of which defines when to and how to invoke a particular (regular or symbiotic) behavior. A behavior policy is encoded as a gene. In SymbioticSphere, evolution occurs on behavior policies (i.e., genes) via genetic operations such as mutation and crossover, which alter behavior policies when agents/platforms replicate themselves or reproduce their offspring. This evolution process is intended to increase the adaptability of agents/platforms by allowing them to adjust their behavior policies to dynamic network conditions across generations. Evolution also frees data center designers from anticipating all possible network conditions and tuning their agents and platforms to the conditions at design time. Instead, agents and platforms can evolve and autonomously adapt themselves to network conditions. This can significantly simplify the implementation and maintenance of agents/platforms.

This paper describes the biologically-inspired mechanisms in SymbioticSphere and evaluates their impacts on the adaptability of data centers. Simulation results show that agents and platforms autonomously adapt to dynamic network conditions (e.g., user location, network traffic and resource availability) by evolving their regular behavior policies. Simulation results also show that agents and platforms coevolve to improve their adaptability by cooperatively adjusting their symbiotic behavior policies.

II. Design Principles in SymbioticSphere

SymbioticSphere applies the following biological principles to design agents and platforms.

(1) Decentralization: In various biological systems (e.g., bee colony), there are no central leader entities to control or coordinate individual entities in order to increase scalability and survivability. Similarly, in SymbioticSphere, there are no central entities to control and coordinate agents/platforms so that they can be scalable and survivable by avoiding a single point of performance bottlenecks [3] and failures [4].

(2) Autonomy: Inspired by biological entities (e.g., bees), agents/platforms sense their local network conditions, and based on the conditions, they autonomously behave and interact with each other without any intervention from/to other agents, platforms and human users.

(3) Emergence: In biological systems, collective (group) behaviors emerge from interactions of individual entities. In SymbioticSphere, agents/platforms interact only with nearby peers. Desirable system characteristics (e.g., adaptability) emerge through collective behaviors and interactions of individual agents/platforms. Note that they are not present in any single agent/platform.

(4) Lifecycle and Food Chain: Biological entities strive to seek and consume food for living. In SymbioticSphere, agents/platforms 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 periodically evaporates energy. The abundance or scarcity of stored energy in agents/platforms affects their lifecycle. For example, an abundance of stored energy indicates high demand to an agent/platform; thus, the agent/platform may be designed to favor reproduction or replication to increase its availability. A scarcity of stored energy indicates lack of demand; it causes death of the agent/platform.

Also, in the ecosystem, the energy accumulated from food is transferred between different species to balance their populations. For example, producers (e.g., shrubs) convert the Sun light energy to chemical energy. The chemical energy is transferred to consumers (e.g., hares) as consumers consume producers [5] (Fig. 1). In SymbioticSphere, the energy exchange among users, agents and platforms is designed after ecological food chain (Fig. 1). SymbioticSphere models a user as the Sun, agents as producers, and platforms as consumers. Similar to the Sun, users have an unlimited amount of energy. The users provide energy to agents in proportion of the services that the users require. When a user requests a service implemented by an agent, the user transfers a certain amount of energy to the agent. (Each agent specifies the price in energy units of its service.). Each agent gains energy from users and transfers 10% of its energy level to the underlying platform for consuming resources provided by the platform. Each platform gains energy from agents and periodically evaporates 10 % of its energy level to the environment. This energy exchange rule follows an ecological fact that a consumer species acquires about 10% of the energy maintained by a producer species [5].

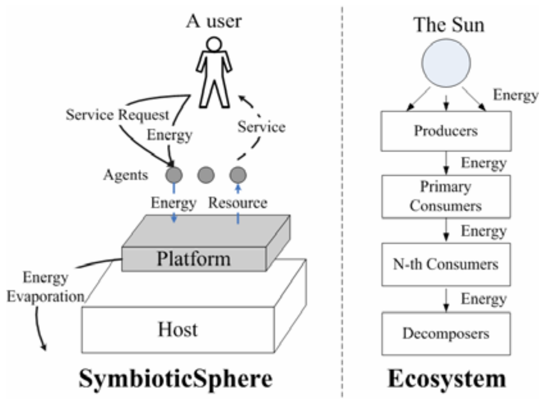

Figure 1. Energy Exchange in SymbioticSphere and Ecosystem

(5) Evolution: Biological entities evolve as a species so that the entities that fit better to the environment become more abundant [6]. In SymbioticSphere, agents and platforms evolve their genes (i.e., behavior policies) by generating behavioral diversity and executing natural selection. Behavioral diversity means that different agents/platforms possess different behavior policies. This is generated via mutation and crossover during replication and reproduction. Natural selection is triggered with agents' and platforms' energy levels. It retains the agents whose energy levels are high (i.e., the agents that have effective behavior policies, such as moving toward a user to gain more energy) and eliminates the agents whose energy levels are low (i.e., the agents that have ineffective behavior policies, such as moving too often). Through successive generations, effective behavior policies become abundant in an agent/platform species while ineffective ones become dormant or extinct. This allows agents/platforms to adapt to dynamic network conditions.

(6) Symbiosis through Coevolution: Competition for food and terrain always occurs in the biological world; however, several species coevolve and establish mutual relationships to avoid excessive competition and support with each other to survive [7]. In SymbioticSphere, agents and platforms cooperate as different species working at different network layers (i.e., application layer and middleware layer) in order to pursue their mutual benefits (e.g., gaining more energy to survive) and improve their adaptability. This is driven by coevolution between agents and platforms, which cooperatively evolves behavior policies for symbiotic behaviors.

III. SymbioticSphere

A. Agents

Each agent consists of three parts: *attributes*, *body* and *behaviors*. *Attributes* carry descriptive information on an agent, such as its ID, energy level, description of a service it provides, and price (in energy units) of the service it provides. *Body* implements a service that an agent provides. For example, an agent may implement a web service, while another may implement a physical model for scientific simulations. *Behaviors* (regular behaviors) implement actions that are inherent to all agents:

- *Replication*: Agents may make a copy of themselves. A replicated (child) agent is placed on the platform that its parent agent resides on, and it inherits the half amount of the parent's energy level.
- *Reproduction*: Agents may produce their offspring with their mating partners. A child agent is placed on the platform that its parent[1] agent resides on, and it receives the half amount of the parent's energy level.
- *Death*: Agents die due to energy starvation. When an agent dies, its underlying platform removes the agent and releases all resources allocated to the agent.
- *Migration*: Agents may move from one platform to another.

B. Platforms

Each platform runs on a host and operates agents (Fig. 1). It consists of *attributes*, *behaviors* and *runtime services*. *Attributes* carry descriptive information on the platform, such as platform ID, energy level and health level.

Health level is defined as a function of three properties: the resource availability on, the age of and the freshness of an underlying host. Resource availability indicates how much resources are available for agents and platforms on a host. Age indicates how long a host has been alive (i.e., how much stable the host is). Freshness indicates how recently a host joined the network. Once a host joins the network, its freshness gradually decreases from the maximum. When an unstable host resumes from a failure, its freshness starts with the value that the host had when it went down. Using age and freshness, unstable hosts and new hosts can be distinguished (Table I).

[1] The parent is an agent/platform that invokes the reproduction behavior.

TABLE I.
FRESHNESS AND AGE IN DIFFERENT TYPES OF HOSTS

Host Type	Freshness	Age
Unstable Host	Lower	Lower
New Host	Higher	Lower
Stable Host	Lower	Higher

Behaviors (regular behaviors) are the actions inherent to all platforms:

- *Replication*: Platforms may make a copy of themselves. A replicated (child) platform is placed on a neighboring host that does not run a platform. (Since there is only one type of platform, two or more platforms are not allowed to run on each host.) It inherits the half of the parent's energy level.
- *Reproduction*: Platforms may produce their offspring with their mating partners. A child platform is placed on a neighboring host that does not run a platform. It inherits the half of the parent's[1] energy level.
- *Death*: Platforms die due to lack of energy. A dying platform kills agents running on it, uninstalls itself and releases all resources the platform uses. Despite the death of a platform, its underlying host remains active so that another platform can run on it in the future.

Runtime services are the middleware services that agents and platforms use to perform their behaviors.

C. Regular Behavior Policies

Regular behavior policies are the behavior policies that each agent/platform has for its regular behaviors. Each policy consists of *factors* (F_i), which evaluate network conditions (e.g., network traffic) or agent/platform status (e.g., energy/health level). Each factor is given a *weight* (W_i). A behavior is invoked if the weighted sum of corresponding factor values ($\Sigma F_i{*}W_i$) exceeds a threshold.

The factors for the agent migration behavior are:

- *Energy Level*: Agent energy level, which encourages agents to move in response to their high energy level.
- *Health Level Ratio*: The ratio of health level on a neighboring platform to the local platform, which encourages agents to move to healthier platforms. This ratio is calculated with three health level properties (HLPs; resource availability, freshness and age):

$$\textit{Health Level Ratio} = \sum_{i=1}^{3}\left(\frac{HLP_i \textit{ on a neighboring platform/host} - HLP_i \textit{ on the local platform/host}}{HLP_i \textit{ on the local platform/host}}\right) \quad (1)$$

- *Service Request Ratio*: The ratio of the number of incoming service requests on a neighboring platform to the local platform. This factor encourages agents to move toward users.

- *Migration Interval*: Time interval to perform migration, which discourages agents to migrate too often.

If there are multiple neighboring platforms that an agent can migrate to, the agent calculates the weighted sum of the above factors for each neighboring platform, and moves to a platform that generates the highest sum.

The factors for the agent reproduction behavior are:

- *Energy Level*: Agent energy level, which encourages agents to reproduce their offspring in response to their high energy levels.
- *Request Queue Length*: The length of a queue, which the local platform stores incoming service requests. This factor encourages agents to reproduce their offspring in response to high demands for their services.

When the weighted sum of the above factor values exceeds a threshold, an agent seeks a mating partner from the local and neighboring platforms. If a mating partner is found, the agent invokes the reproduction behavior. Otherwise, the agent invokes the replication behavior. Section III.F describes how an agent seeks its mating partner.

The factors for the agent death behavior are:

- *Energy Level*: Agent energy level. Agents die when they run out of their energy.
- *Energy Loss Rate*: The rate of energy loss, calculated with Eq. (2). E_t and E_{t-1} denote the energy levels in the current and previous time instants. Agents die in response to sharp drops in demands for their services.

$$Energy\ \ Loss\ \ Rate = \frac{E_{t-1} - E_t}{E_{t-1}} \qquad (2)$$

The factors for the platform reproduction behavior are:

- *Energy Level*: Platform energy level, which encourages platforms to reproduce their offspring in response to their high energy levels.
- *Health Level Ratio*: The ratio of health level on a neighboring host to the local host. This factor encourages platforms to reproduce their offspring on the hosts that generate higher values with Eq. (1).
- *The Number of Agents*: The number of agents working on each platform. This factor encourages platforms to reproduce their offspring in response to high agent population on them.

When the weighted sum of the above factor values exceeds a threshold, a platform seeks a mating partner from its neighboring hosts. If a mating partner is found, the platform invokes the reproduction behavior. Otherwise, it invokes the replication behavior. Section III.F describes how a platform finds its mating partner. If there are multiple neighboring hosts that a platform can place its child platform on, it places the child on a host whose health ratio is highest among others.

The factors for the platform death behavior are:

- *The Number of Agents*: The number of agents running on each platform. This factor discourages platforms to die when agents run on them.
- *Energy Loss Rate*: The rate of energy loss, calculated with Eq. (2). Platforms die in response to sharp drops in demands for their resources.

Each agent/platform expends energy to invoke behaviors (i.e., behavior cost) except the death behavior. When the energy level of an agent/platform exceeds the cost of a behavior, it decides whether it performs the behavior by calculating a weighted sum described above.

D. Symbiotic Behaviors

Each symbiotic behavior is defined as a sequence of regular behaviors that an agent and its underlying platform perform in order. There are two types of symbiotic behaviors: *agent-initiated* symbiotic behaviors (A1, A2 and A3 behaviors) and *platform-initiated* symbiotic behaviors (P1, P2 and P3 behaviors) as described below.

A1: When an agent wants to move toward a user but there is no platform running on a neighboring host closer to the user, the agent may propose the local platform to replicate itself on the neighboring host (Fig. 2). If the local platform's health level is low, the platform accepts the agent's proposal. The agent gives the platform the energy units of platform replication cost, and the platform replicates itself on a host that the agent wants to migrate to. As a result, the agent can migrate to the child platform and improve response time. The platform can improve its health level because resource availability becomes higher.

A2: When an agent is dying due to energy starvation, the agent may ask the local platform to shoulder agent migration cost so that it can migrate to a platform on a healthier platform (i.e., a platform less crowded with agents) (Fig. 3). If the local platform's health level is low, it agrees with the agent. As a result, the agent can have a chance to receive more service requests (i.e., energy) and survive longer. The platform can improve its health level because resource availability becomes higher.

A3: When an agent is dying due to energy starvation, the agent may ask the local platform to shoulder agent migration cost so that the agent can migrate to a neighboring platform closer to a user (Fig. 4). If the local platform's health level is low, the platform agrees with the agent. As a result, the agent can improve response time. The platform can improve its health level because resource availability becomes higher.

P1: When a platform replicates itself on a neighboring host, the platform may propose an agent working on it to migrate to a replicated (child) platform (Fig. 5). If the agent's energy level is low, it accepts the platform's proposal. The platform provides the agent with the energy units of agent migration cost. As a result, the platform can increase its health level because resource availability becomes higher. A child platform can survive longer because it gains energy from the migrating agent. On its destination platform (i.e., a platform less crowded with agents), the agent can have a chance to receive more service requests (i.e., energy) from users and survive longer.

P2: When a platform's health level is low, the platform may propose an agent working on it to migrate to a healthier neighboring platform (Fig. 6). If the agent's energy level is low, it accepts the platform's proposal.

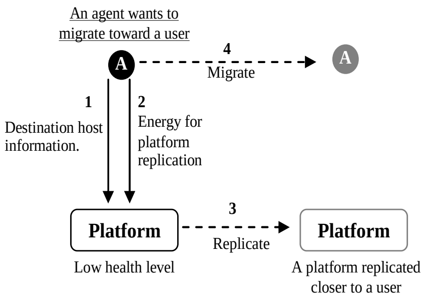
An agent wants to migrate toward a user
A
4
Migrate
A
1
Destination host information.
2
Energy for platform replication
Platform
3
Replicate
Platform
Low health level
A platform replicated closer to a user

Figure 2. Symbiotic Behavior A1

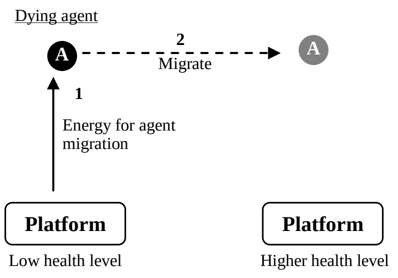

Figure 3. Symbiotic Behavior A2

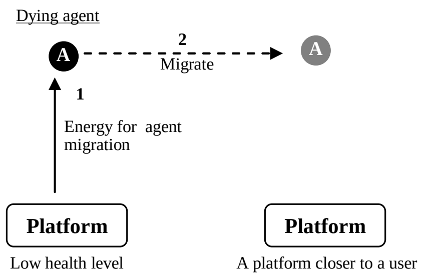

Figure 4. Symbiotic Behavior A3

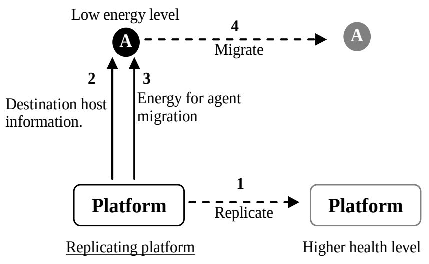

Figure 5. Symbiotic Behavior P1

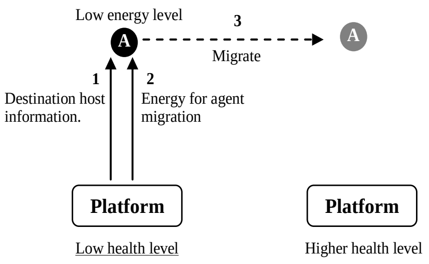

Figure 6. Symbiotic Behavior P2

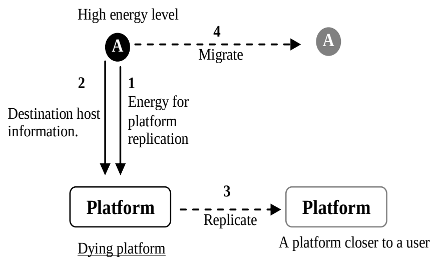
High energy level
A
4
Migrate
A
2
Destination host information.
1
Energy for platform replication
Platform
3
Replicate
Platform
Dying platform
A platform closer to a user

Figure 7. Symbiotic Behavior P3

The platform provides the agent with the energy units of agent migration cost. As a result, the platform can increase its health level because resource availability becomes higher. The migrating agent can avoid working on an unhealthy platform that may crash, and have a chance to receive more service requests (i.e., energy) from users and survive longer on a healthier platform (i.e., a platform less crowded with agents).

P3: When a platform is dying due to energy starvation, the platform may ask the local agents to shoulder platform replication cost so that it can replicate itself on a host closer to a user (Fig. 7). If the platform dies, the agents die off on the platform. Thus, some of them accept the platform's proposal if their energy level is high. As a result, the migrating agents can avoid death and gain more energy from a user and survive longer on their destination platform. A child platform can secure energy intake from the migrating agents and survive longer.

E. Symbiotic Behavior Policies

A symbiotic behavior policy is a behavior policy that each agent/platform possesses to determine whether it invokes a particular symbiotic behavior. Each symbiotic behavior policy consists of a *proposer policy* and an *acceptor policy*. A proposer policy determines when one species (e.g., an agent) proposes a symbiotic behavior to the other species (e.g., a platform). An acceptor policy determines when one species (e.g., a platform) accepts a proposal that the other species (e.g., an agent) makes to invoke a symbiotic behavior.

Each proposer policy consists of a *proposer's precondition* and a *factor* (F^P) (Table II.). A proposer prepares to propose a symbiotic behavior when its precondition is satisfied. For example, an agent prepares to propose the A1 behavior when it wants to move toward a user but there is no platform running on a neighboring host closer to the user. A platform prepares to propose the P3 behavior when it is dying (i.e., when its weighted sum of the death behavior factors exceeds a threshold). Each F^P (proposer policy factor) represents a proposer's status such as energy/health level. Each factor is given a *weight* (W^P). As far as a proposer's precondition is satisfied, the proposer proposes a symbiotic behavior if a corresponding weighted factor value (F^P*W^P) is below a threshold.

TABLE II.
PROPOSER POLICIES

Proposer	Symbiotic Behavior	Factor (F^P)	Proposer's Precondition
Agent	A1	1/Energy Level	Wants to migrate?
	A2	Energy Level	Is dying?
	A3	Energy Level	Is dying?
Platform	P1	1/Energy Level	Replicating?
	P2	Health Level	N/A
	P3	Energy Level	Is dying?

Each acceptor policy contains a *factor* (F^A), which evaluates the acceptor's status such as energy level and health level (Table III). Each factor is given a *weight* (W^A). When an acceptor receives a proposal that a proposer makes to invoke a symbiotic behavior, the acceptor calculates the weighted factor value (F^A*W^A) for the symbiotic behavior. If it is below a threshold, the acceptor accepts the proposal. Once a proposal is accepted, a proposer initiates a symbiotic behavior.

TABLE III.
ACCEPTOR POLICIES

Acceptor	Symbiotic Behaviors	Factor (F^A)
Platform	A1	Health Level
	A2	Health Level
	A3	Health Level
Agent	P1	Energy Level
	P2	Energy Level
	P3	1/Energy Level

F. Evolutionary Process

The weight and threshold values in behavior policies have significant impacts on the adaptability of agents and platforms. However, it is hard to anticipate all possible network conditions and find an appropriate set of weight and threshold values for the conditions. As shown in Sections III.C and III.E, there are 42 weight and threshold values in total (18 for regular behaviors and 24 for symbiotic behaviors). Assuming that 10 different values can be assigned to each weight and threshold, there are 10^{42} possible combinations of weight and threshold values.

Instead that data center designers manually assign weight and threshold values, SymbioticSphere allows agents and platforms to autonomously find appropriate values in an evolutionary manner, thereby adapting themselves to network conditions. Both regular and symbiotic behavior policies are encoded as genes of agents and platforms. Each gene contains one or more weight values and a threshold value for a particular behavior.

For regular behaviors, each agent/platform has a gene (i.e., a set of weight and threshold values) for each behavior. Figs. 8 and 9 show the gene structure for agent/platform behaviors. For example, for the agent reproduction behavior, a gene is structured to have three elements: (1) Wr_1, a weight value for the energy level factor; (2) Wr_2, a weight value for the factor of request queue length; and (3) Tr, a threshold value (Fig. 8).

Each weight value is a decimal number in the range of [0…1], and it is initialized randomly. Each threshold value is a decimal number in the range of [0…M], where M denotes the number of considered factors. Each threshold value is also initialized randomly. For example, the agent reproduction behavior has a threshold value in the range of [0...2] because the behavior considers two factors: Wr_1 and Wr_2 (see Section 3.C and Fig. 8).

Reproduction Policy			Migration Policy					Death Policy		
Wr_1	Wr_2	**Tr**	Wm_1	Wm_2	Wm_3	Wm_4	**Tm**	Wd_1	Wd_2	**Td**

Figure 8. Gene Structure for Agent Regular Behaviors

Reproduction Policy				Death Policy		
Wr_1	Wr_2	Wr_3	**Tr**	Wd_1	Wd_2	**Td**

Figure 9. Gene Structure for Platform Regular Behaviors

For symbiotic behaviors, each agent/platform has a gene (i.e., a set of weight and threshold values) for each symbiotic behavior. Fig. 10 shows the gene structure of agents for symbiotic behaviors. The structure consists of the genes for agent-initiated symbiotic behaviors (i.e., proposer policy) and the genes for platform-initiated symbiotic behaviors (i.e., acceptor policy). For example, for the A1 symbiotic behavior, each agent has a gene consisting of two elements: (1) W^P_{A1}, a weight value used to examine whether or not to propose the A1 behavior; and (2) T^P_{A1}, a threshold value. Fig. 11 shows the gene structure of platforms for symbiotic behaviors. The structure consists of the genes for platform-initiated symbiotic behaviors (i.e., proposer policy) and the genes for agent-initiated symbiotic behaviors (i.e., acceptor policy). For example, for the A1 symbiotic behavior, each platform has a gene consisting of two elements: (1) W^A_{A1}, a weight value used to examine whether or not to accept a proposal on the A1 behavior; and (2) T^A_{A1}, a threshold value. Each weight and threshold value is a decimal number in the range of [0…1], and it is initialized randomly.

Proposer Policy						Accepter Policy					
W^P_{A1}	T^P_{A1}	W^P_{A2}	T^P_{A2}	W^P_{A3}	T^P_{A3}	W^A_{P1}	T^A_{P1}	W^A_{P2}	T^A_{P2}	W^A_{P3}	T^A_{P3}

Figure 10. Gene Structure of Agents for Symbiotic Behaviors

Accepter Policy						Proposer Policy					
W^A_{A1}	T^A_{A1}	W^A_{A2}	T^A_{A2}	W^A_{A3}	T^A_{A3}	W^P_{P1}	T^P_{P1}	W^P_{P2}	T^P_{P2}	W^P_{P3}	T^P_{P3}

Figure 11. Gene Structure of Platforms for Symbiotic Behaviors

The genes of agents and platforms are altered via genetic operations (genetic crossover and mutation) when they perform the reproduction and replication behaviors. As described in Sections III.A and III.B, each agent/platform selects a mating partner when it performs the reproduction behavior. A mating partner is selected by ranking agents/platforms running on the local and neighboring hosts. For this ranking process, SymbioticSphere uses a domination ranking mechanism [8].

Agents and platforms are ranked with two objectives: (1) energy utility (Eq. (3)) and (2) behavior invocation efficiency (Eq. (4)). Behavior invocation efficiency indicates how an agent/platform behaves in an energy efficient manner. In both objectives, the higher, the better.

$$Energy\ Utility = 1 - \frac{Total\ Energy\ Expenditure}{Total\ Energy\ Gain} \tag{3}$$

$$Behavior\ Invocation\ Efficiency = \frac{Total\ Energy\ Gain}{Total\ \#\ of\ Behavior\ Invocations} \tag{4}$$

Agents/platforms are plotted on a two dimensional space whose axes are the objectives described above. Fig. 12 shows an example to rank four different agents (Agent *A* to *D*). In this example, Agent *A* dominates the other three agents in both of two objectives. (In other words, Agent *A* is non-dominated.) Therefore, the agent is given Rank 1. Agent *B* is dominated by Agent *A*; however, it dominates the other two agents (Agent *C* and *D*). Thus, Agent *B* is given Rank 2. Agent *C* and *D* are dominated by Agent *B*, and they cannot dominate with each other. Thus, they are given Rank 3.

During reproduction, an agent/platform ranks other agents/platforms running on the local and neighboring hosts, as described above, and selects the one in the highest rank as a mating partner. If the parent agent/platform, which invokes the reproduction behavior[1], is in the highest rank, it fails to find its mating partner and performs the replication behavior.

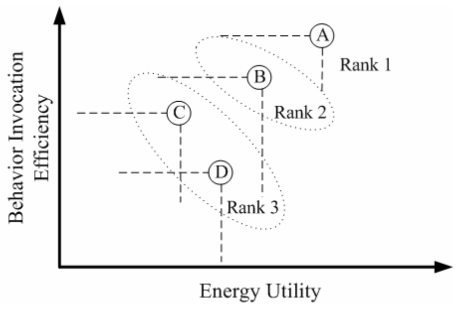

Figure 12. An Example Domination Ranking

In reproduction, a genetic crossover occurs. A parent and its mating partner contribute their genes and randomly combine them for a child's gene (Fig. 13). Then, a genetic mutation occurs on the child's gene. Each gene element (i.e., weight or threshold value) is randomly altered with a mutation probability (Fig. 13).

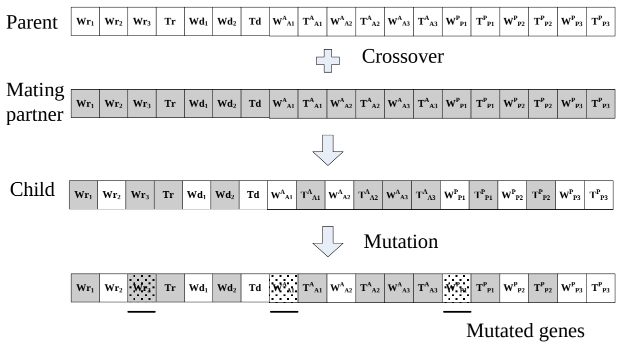
Parent
Crossover
Mating partner
Child
Mutation
Mutated genes
Wr1 Wr2 Wr3 Tr Wd1 Wd2 Td WA A1 TA A1 WA A2 TA A2 WA A3 TA A3 WP P1 TP P1 WP P2 TP P2 WP P3 TP P3

Figure 13. Example Genetic Operations

In replication, a parent copies its gene to its child. Then, a mutation occurs on the child's gene in the same way as the mutation in reproduction.

IV. Simulation Results

This section shows a set of simulation results to evaluate how agents and platforms adapt to dynamics of the network by using their regular and symbiotic behaviors through evolution. Section IV.A evaluates how regular behaviors impact the adaptability of agents and platforms. Section IV.B evaluates how symbiotic behaviors improve the adaptability of agents and platforms. Section IV.C demonstrates how agents and platforms evolve their behavior policies (genes). The same set of simulation configurations is used for all of the three experiments. Simulations were carried out with the SymbioticSphere simulator[2], which implements the biologically-inspired mechanisms described in Section III.

Figure 14 shows a simulated network. A data center operates on the network, and consists of hosts connected in a 7x7 grid topology. Users send service requests to agents via user access point. This paper assumes that a single (virtual) user runs on the access point, and it emulates multiple users to send service requests. At the beginning of each simulation, one agent and one platform are deployed on a host that is farthest from the user.

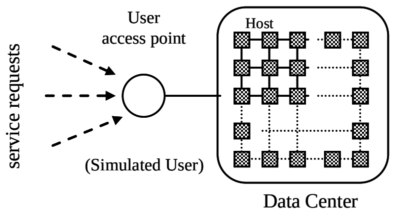

Figure 14. Simulated Network

Each host has 256 MB memory space[3]. Of the space, an operating system and a Java VM consume 128 and 64 MB, respectively. The remaining space is available for a platform and agents on each host. Each agent and platform consumes 5 and 20 MB, respectively. This assumption is obtained from a prior empirical experiment [9].

A host operates in the active or inactive state. When a platform works on a host, the host is active and consumes 60W power. The host becomes inactive when a platform dies on it. An inactive host consumes 5W power. This assumption on power consumption is obtained from [10]. A host is assumed to become active from the inactive state using the Wake On LAN (WOL) technology [11]. When a platform replicates itself on an inactive host, the platform sends a WOL packet to the host to wake it up.

Figure 15 shows how the user changes service request rate over time. This service request rate is taken from a workload trace of the 1998 Olympic official website [12]. The peak workload is 9,600 requests/min.

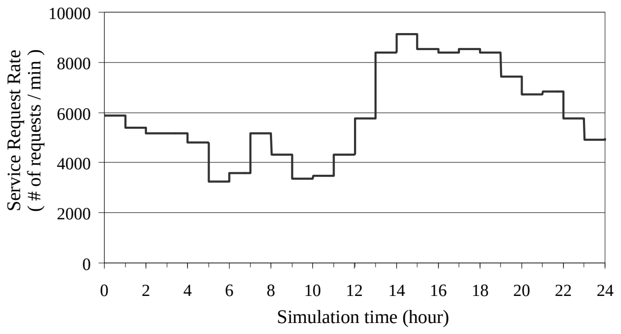

Figure 15. Service Request Rate

Figure 16 shows a pseudo code to run the user, agents and platforms in each simulation cycle.

In this paper, adaptability is defined as *service adaptation* and *resource adaptation*. Service adaptation represents the quality and availability of services provided by agents. The quality of services is measured as response time for the user. Service availability is measured as the number of agents. Resource adaptation represents resource availability and resource efficiency. Resource availability is measured as the number of platforms that makes resources available for agents. Resource efficiency indicates how many service requests are processed per resource utilization of agents and platforms. It is measured as (the total number of service requests processed by

[2] The current code base of the SymbioticSphere simulator contains 15,100 lines of Java code. This simulator is freely available at http://dssg.cs.umb.edu/projects/SymbioticSphere/.

[3] In this paper, memory availability represents resource availability.

```
While( not simulation last cycle )
  The user sends service requests to agents according to a certain rate.
  For each agent Do
    If ( a service request(s) received )
      Process the request(s) and gain energy.
    End If
    Decide whether or not to invoke regular behaviors.
    Decide whether or not to invoke symbiotic behaviors.
    Expend energy to the local platform.
  End For
  For each platform Do
    Decide whether or not to invoke regular behaviors.
    Decide whether or not to invoke symbiotic behaviors.
    Update health level.
    Evaporate energy.
  End For
End While
```

Figure 16. Pseudo Code of Simulation Cycle

agents) / (the total amount resources consumed by agents and platforms).

A. Evaluation of Regular Behaviors

This section evaluates how agents and platforms autonomously adapt to dynamic network conditions by using regular behaviors except the reproduction behavior. For the experiments in this section, no agents and platforms invoke symbiotic behaviors and perform evolutionary process. Rather, agents and platforms use the regular behavior policies that the authors manually configured through trial and errors. (The authors spent approximately 170 hours for the trial and errors.)

The simulation results in this section are compared with the analytically optimal results. Fig. 17 shows an analytical model used in this paper. The analytical model is written in AMPL [14], and designed to generate the number of agents (*a*), the number of platforms (*p*), response time (*responseTime*) and resource efficiency (*resoureEff*), while minimizing response time and maximizing resource efficiency. Since this analytical model is not linear, it is solved with MINOS (Modular In-core Nonlinear Optimization System) solver [15].

```
var a >=1, integer;        # the number of agents
var p >=1, integer;        # the number of platforms
var N >=1,  integer;      # the number of columns in a mesh network
var M >=1, integer;       # the number of rows in a mesh network
param requests := 9600; # the number of  user requests (input)
var responseTime;
var resourceEff;
    minimize objective:  0.5*responseTime-0.5*resourceEff;
    subject to
            responseTime = (requests/(10*a)) + ( 0.01 * (N+M)/3 );
            resourceEff = requests/(20*p + 5*a);
            p =   ceil ( a/8 );
            p <= 49;
            N =  ceil (p/ 7);
            M =  floor ( p / N);
            M*N <=49;
            responseTime  <= 3;
```

Figure 17. An Analytical Model to Evaluate Service and Resource Adaptation

In this analytical model, response time includes the processing overhead for an agent to process a service request and the transmission latency of the request between the agent and user. The processing overhead is calculated with the number of service requests, the number of available agents, and the number of service requests that an agent can process in one second. (Each agent processes a service request in 0.1 second; therefore, it can process 10 requests in one second). The transmission latency is calculated with the average hop count between an agent and the user[4] and the time required for a service request to travel between two nodes (0.01 sec.). The (optimal) number of platforms is calculated with the number of available agents and the maximum number of agents that can work on each platform. (This maximum number is eight.) Throughput is calculated as the ratio of the number of service requests processed by agents to the number of given service requests. It is assumed to be 100% when response time is less than three seconds.

Fig. 18 shows how service availability (i.e., the number of agents) and resource availability (i.e., the number of platforms) change dynamically. Starting with one agent and one platform deployed at 0:00, they change their populations through replication in order to process the demand placed on them (6,000 requests/min; See Fig. 15.). When service request rate increases from 12:00 to 14:00, agents gain more energy from the user and replicate themselves more often. In response to higher energy intake, they also transfer more energy to platforms. As a result, platforms also increase their population through replication. When service request rate decreases at 15:00, some agents and platforms die because they cannot balance their energy gain and expenditure due to less energy transfer from the user. Fig. 18 shows that agents and platforms autonomously adapt their availability to dynamic demand changes with their regular behaviors.

Fig. 18 also shows the analytical results on the number of agents/platform. Between analytical and simulation results, differences are 15% and 21% on average for the number of agents platforms, respectively.

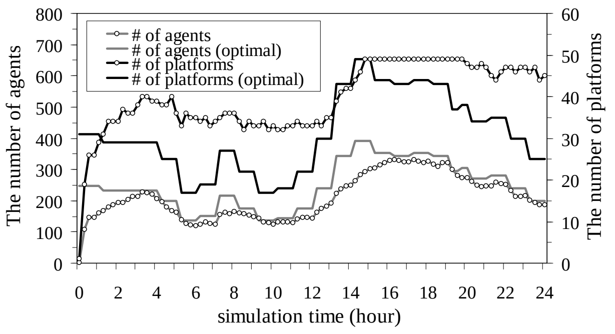

Figure 18. The Number of Agents and Platforms

Fig. 19 shows the average response time (i.e., the quality of services) and throughput of agents. At the beginning of a simulation, response time is high (over 20 seconds) because there exists only one agent and one platform to process 6,000 requests a minute. Thus, throughput does not reach 100%. However, the agent and platform immediately replicate themselves, and agents mi-

[4] Assuming a mesh network, the average hop count is (N+M)/3, where N and M denotes the number of columns and rows, respectively [13].

grate toward the user. As a result, response time drops below five seconds at 1:00, and throughput reaches 100%. After 1:00, response time is constantly around five seconds, and throughput is constantly almost 100%. When service request rate increases from 12:00 to 14:00, response time spikes. However, agents decrease response time again through replication. See also Fig. 18 for the changes in the number of agents from 12:00 to 14:00. Fig. 19 shows that agents and platforms collectively retain response time and throughput performance by adapting their populations and locations to demand changes.

Fig. 19 also shows the analytical results on the average response time and throughput. The average difference is only 2% between analytical and simulation results. Response time is mostly same after 14:00 between analytical and simulation results.

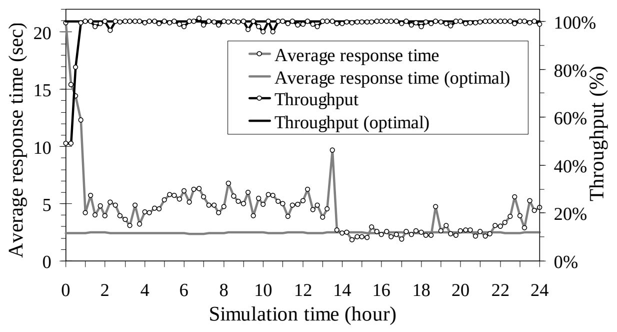

Figure 19. Response Time and Throughput

Fig. 20 shows resource efficiency. Platforms adapt resource efficiency according to demand changes by adjusting their availability. (See also Fig. 18.) On average, difference is only 3% between analytical and simulation results.

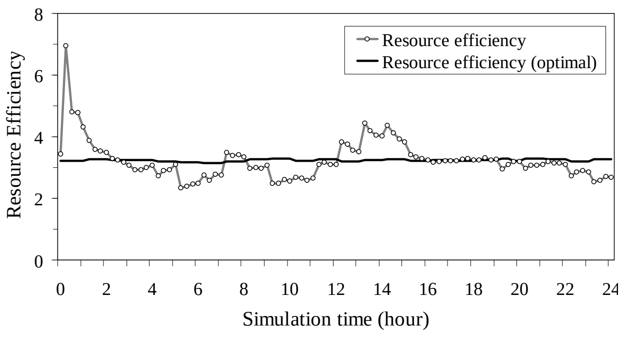

Figure 20. Resource Efficiency

Figs. 18 to 20 show that SymbioticSphere yields the service and resource adaptation results close enough to the analytically optimal results.

B. *Evaluation of Symbiotic Behaviors*

This section evaluates how symbiotic behaviors complement regular behaviors and augment the adaptability of agents and platforms. For the experiments in this section, no agents and platforms perform evolutionary process. Rather, agents and platforms use the regular and symbiotic behavior policies that the authors manually found through trial and errors. (The authors spent approximately 170 hours for the trial and errors.)

Fig. 21 shows that agent-initiated symbiotic behaviors (A1, A2 and A3) contribute to improve response time (i.e., service adaptation), compared with using regular behaviors only. Agents help platforms increase their availability near the user, and platforms help agents move toward the user or to healthier platforms. As a result, agents can process service requests more quickly.

Fig. 22 shows that agent-initiated symbiotic behaviors improve resource efficiency (i.e., resource adaptation), compared with using regular behaviors only. Agents help platforms increase their availability near the user, and platforms help agents move toward the user or to healthier platforms. Then, the availability of platforms decreases on the hosts far from the user due to energy starvation. As a result, agents can process service requests in a timely manner, while platforms can reduce their resource consumption. This allows agents and platforms to cooperatively increase resource efficiency.

Figs. 21 and 22 show that symbiotic behaviors can improve the adaptability of agents and platforms. However, it is not completely clear whether symbiotic behaviors significantly improve response time and resource availability because of high variance. Therefore, this simulation study carried out an ANOVA (analysis of variance) method to evaluate how the response time and resource efficiency results are better in using symbiotic behaviors, compared with using regular behaviors only. The ANOVA results demonstrate that symbiotic behaviors yield better response time and resource efficiency results than regular behaviors with the confidence of 99.99%.

Figs. 23, 24, 25, 26, 27 and 28 show the average results of load balancing, throughput, platform health level, agent energy level, platform energy level and power consumption in both cases to use regular behaviors only and symbiotic behaviors as well as regular behaviors.

Figure 23 shows the average load balancing index (LBI), which indicates how workload (i.e., the number of service requests) is distributed over available platforms. LBI is measured with Eq. (5). It is the standard deviation of workload among platforms.

$$Load\ \ Balancing\ \ Index = \sqrt{\frac{\sum_{i}^{N}(X_i - \mu)^2}{N}} \qquad (5)$$

X_i denotes (the number of messages processed by agents running on platform i) / (the amount of resources utilized by platform i and agents running on platform i). μ is the average of X_i, which means (the total number of messages processed by all agents) / (the total amount of resources utilized by all platforms and all agents). N denotes the number of available platforms. Platform-initiated symbiotic behaviors contribute to improve LBI. Platforms help agents to move to healthier platforms. The workload is distributed over available platforms.

Fig. 24 shows the average throughput of agents. It illustrates that some of symbiotic behaviors contribute to improve agent throughput.

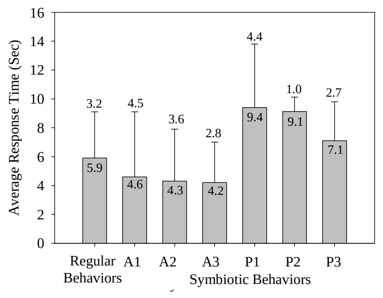

Figure 21. Average Response Time

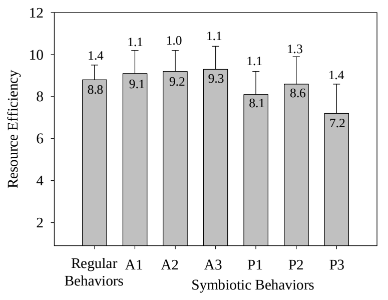

Figure 22. Resource Efficiency

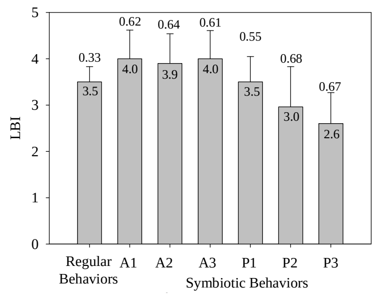

Figure 23. LBI

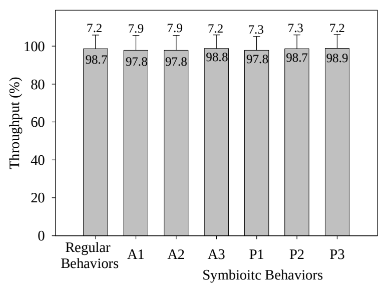

Figure 24. Throughput

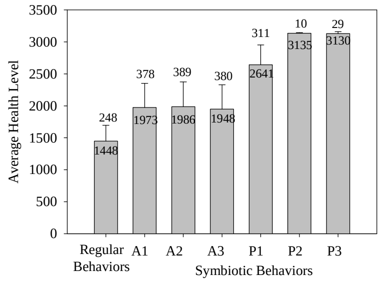

Figure 25. Average Health Level

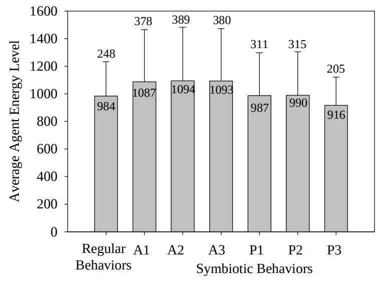

Figure 26. Average Agent Energy Level

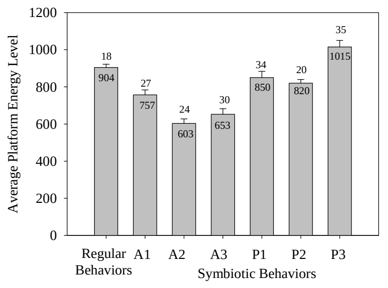

Figure 27. Average Platform Energy Level

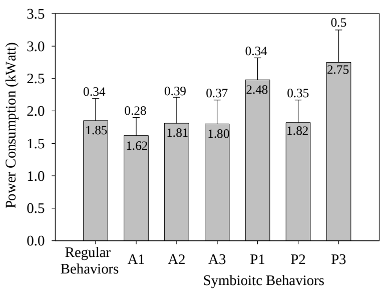

Figure 28. Average Power Consumption

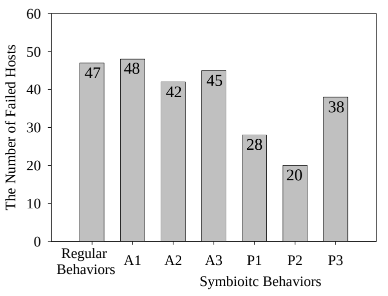

Figure 29. The Number of Failed Hosts

Fig. 25 shows the average health level of platforms. All symbiotic behaviors improve health level, compared with using regular behaviors only. This occurs because platforms help agents move to healthier platforms and agents help platforms replicate on healthier hosts.

Fig. 26 shows the average energy level of agents. Agent-initiated symbiotic behaviors increase agent energy level because platforms help agents move toward the user and gain more energy.

Fig. 27 shows the average energy level of platforms. The P3 behavior increases platform energy level because agents help platforms replicate on hosts closer to the user and agents can migrate to the replicated platforms. This way, agents gain more energy and transfer more energy to platforms.

Fig. 28 shows the average power consumption of agents and platforms. Agent-initiated symbiotic behaviors contribute to save power consumption. Agents help platforms increase their availability near the user, and platforms help agents move toward the user or to healthier platforms. Then, platforms die on the hosts far from the user due to energy starvation. As a result, power consumption is reduced.

Fig. 29 shows the total number of failed hosts during a simulation. This simulation study assumes that each host has 40% probability to crash when available memory space is less than 5MB for 15 minuets[5]. (A failed host resumes in 5 minutes.) Platform-initiated symbiotic behaviors, particularly the P2 behavior, contribute to reduce the number host failures because platforms help agents move to healthier platforms.

Fig. 30 shows how the number of agents changes against a data center failure, where the link between the user access point to a data center fails at 14:00 for five minutes. Agents with regular behaviors immediately die off because energy transfer stops from the user to them due to a link failure. On the other hand, if agents have the A3 behavior, they can cooperate with platforms to move

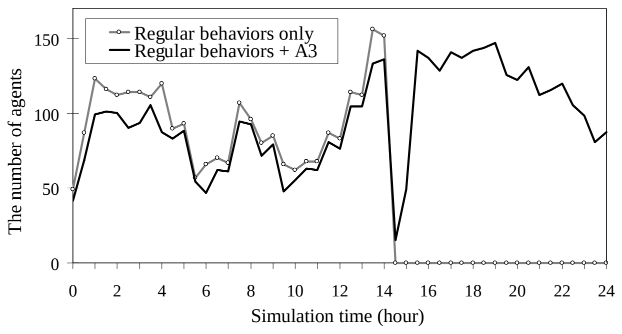

Figure 30. The Number of Agents

[5] Memory utilization in overload is the most common reason for host crashes [16].

to platforms closer to a user so that they gain service requests (i.e. energy) once a failed link resumes. Fig. 30 shows that the A3 behavior contributes for agents to survive link failures and retain service availability.

Fig. 31 shows how agents and platforms perform better when they perform individual (or a combination of) symbiotic behaviors, compared with performing regular behaviors only. The performance of agents and platforms is measured with Eq. (6) using seven metrics: response time, resource efficiency, LBI, throughput, platform health level, agent energy level and platform energy level. PS_i denotes the performance result in terms of the metric i when agents and platforms perform symbiotic behaviors as well as regular behaviors. PR_i denotes the performance result in terms of the metric i when agents and platforms perform regular behaviors only.

$$Performance\ \ Ratio = \sum_{i=1}^{7}\left(\frac{PSi - PRi}{PRi}\right) \qquad (6)$$

Fig. 31 shows that agents and platforms yield better performance ratio when they perform multiple symbiotic behaviors, compared with performing individual symbiotic behaviors. Combinations of symbiotic behaviors allow agents and platforms to perform better in multiple performance metrics simultaneously. For example, P3A1 improves response time and LBI simultaneously because agents can reduce response time with the A1 behavior and platforms can decrease LBI with the P3 behavior.

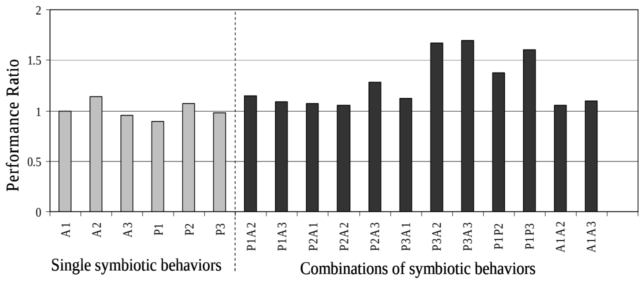

Figure 31. The Performance Ratio

C. Evaluation of Evolution and Coevolution

This section evaluates how evolution and coevolution contribute to the adaptability of agents and platforms. Each simulation was carried out for 10 days by repeating the daily workload trace 10 times. At the beginning of a simulation, an agent and a platform are deployed on each host (49 agents and 49 platforms in total).

Four simulation scenarios are used to evaluate the adaptability of agents and platforms.

Scenario R: This scenario is same as the one used in Section IV.A. Agents and platforms invoke behaviors (except the reproduction behavior), and do not perform evolution. They use the behavior policies that the authors manually configured.

Scenario R+S: Agents and platforms invoke the A3 and P3 symbiotic behaviors as well as regular behaviors (except the reproduction behavior). They invoke the two behaviors because a combination of A3 and P3 yields the best performance in Fig. 30. They do not perform evolution, and use the behavior policies that the authors manually configured.

Scenario RG: This scenario is similar to the scenario R in that agents and platforms invoke regular behaviors. The difference between this scenario and the scenario R is that agents and platforms perform evolution. They use the mutation probability of 0.05.

Scenario 4 (RG+SG): This scenario is similar to the scenario R+S in that agents and platforms invoke the A3 and P3 symbiotic behaviors as well as regular behaviors. The difference between this scenario and the scenario R+S is that agents and platforms perform evolution. They use the mutation probability of 0.05.

Fig. 32 shows how service availability (i.e., the number of agents) changes dynamically in the scenarios RG and RG+SG. Fig. 33 shows the daily average number of agents in the scenarios R, R+S, RG and RG+SG. At the beginning of a simulation, the daily average number of agents fluctuates in the RG and RG+SG scenarios because agents still search appropriate behavior policies by altering their genes through evolution. However, as the time goes, the RG and RG+SG results become more stable and closer to the R and R+S results. This indicates that agents evolve and autonomously adapt their behavior policies to dynamic network conditions.

At Day 10, the daily average number of agents is higher in the scenario R+S than the scenario R, and it is also higher in the RG+SG scenario than the scenario RG. This demonstrates that symbiotic behaviors (A3 and P3) allow agents to gain more energy from the user and survive longer, thereby retaining higher service availability.

Fig. 34 shows how resource availability (i.e., the number of platforms) changes dynamically in the scenario RG and RG+SG. Fig. 35 shows the daily average number of platforms in the scenarios R, R+S, RG and RG+SG. At the beginning of a simulation, the daily average number of platforms fluctuates in the RG and RG+SG scenarios because platforms still search appropriate behavior policies by altering their genes through evolution. However, as the time goes, the RG and RG+SG results become more stable and closer to the R and R+S results. This indicates that platforms evolve and autonomously adapt their behavior policies to dynamic network conditions.

At Day 10, the daily average number of platforms is higher in the scenario R+S than the scenario R, and it is also higher in the RG+SG scenario than the RG scenario. This demonstrates that symbiotic behaviors allow platforms to gain more energy from agents and survive longer, thereby retaining higher resource availability.

Fig. 36 shows how average response time (i.e., the quality of services) changes dynamically in the scenario RG and RG+SG. Fig. 37 shows the daily average response time in the scenarios R, R+S, RG and RG+SG. At the beginning of a simulation, agents and platforms still search appropriate behavior policies by altering their genes through evolution. Thus, they do not behave adaptively yet; the RG and RG+SG results are higher and less stable than the RG and RG+SG results. However, the RG and RG+SG results gradually become more stable and closer to the R and R+G results. This indicates that agents

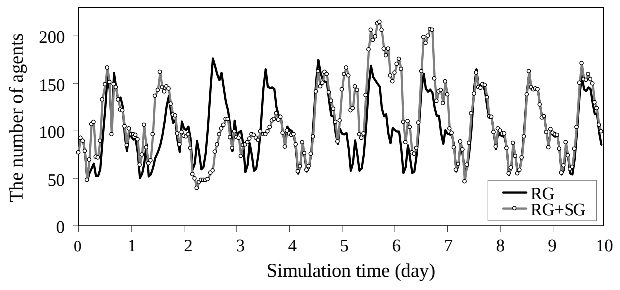

Figure 32 The Number of Agents

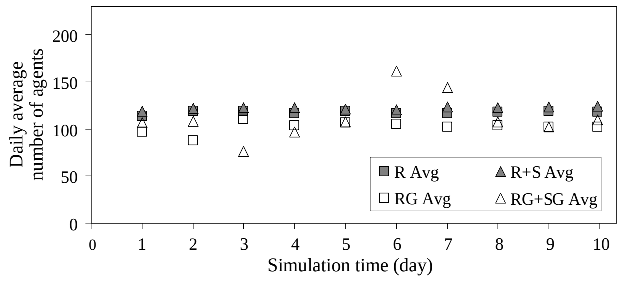

Figure 33 Daily Average Number of Agents

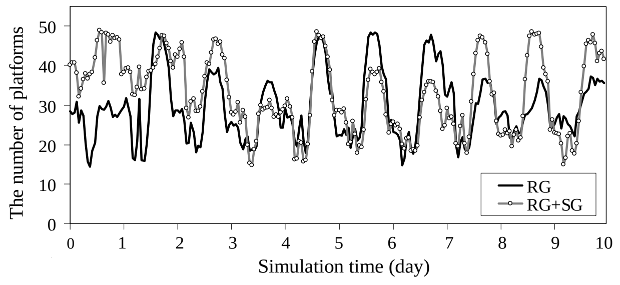

Figure 34 The Number of Platforms

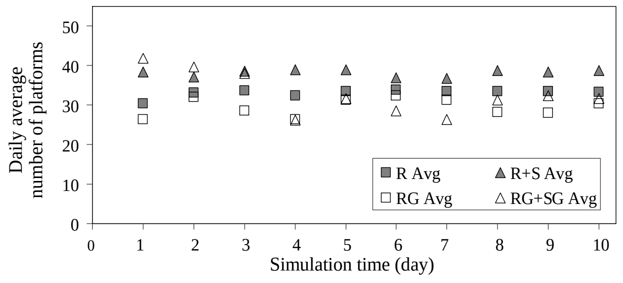

Figure 35 Daily Average Number of Platforms

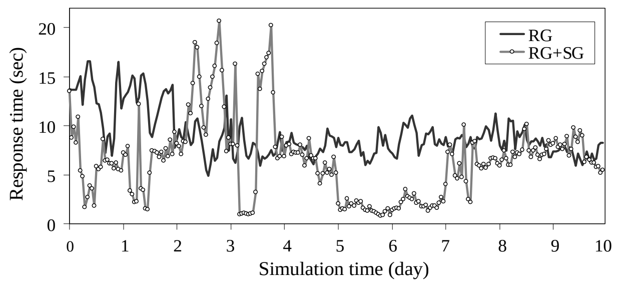

Figure 36 Response Time

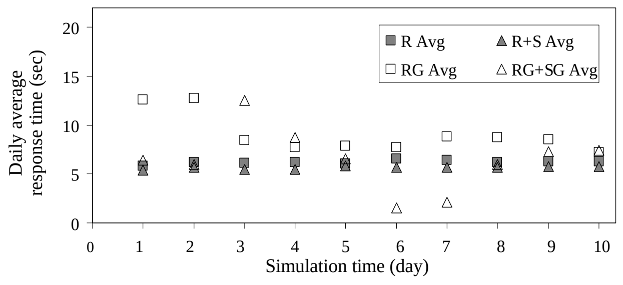

Figure 37 Daily Average Response Time

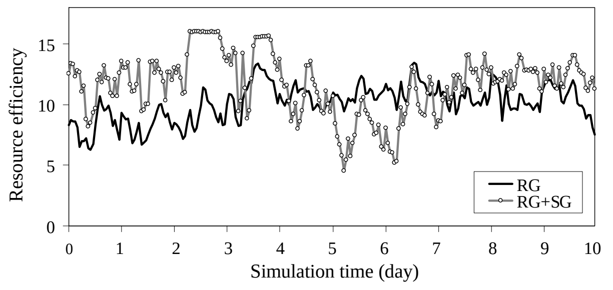

Figure 38 Resource Efficiency

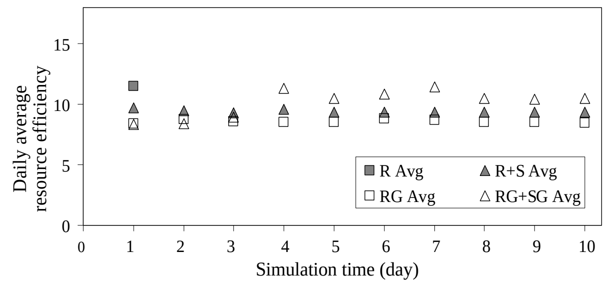

Figure 39 Daily Average Resource Efficiency

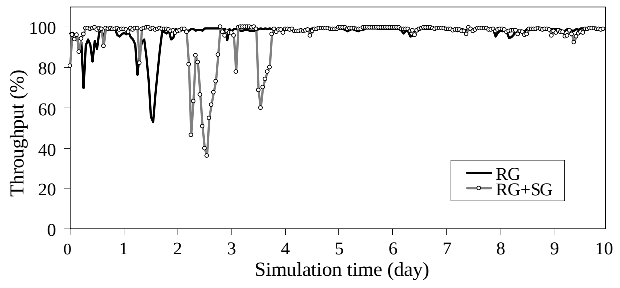

Figure 40 Throughput

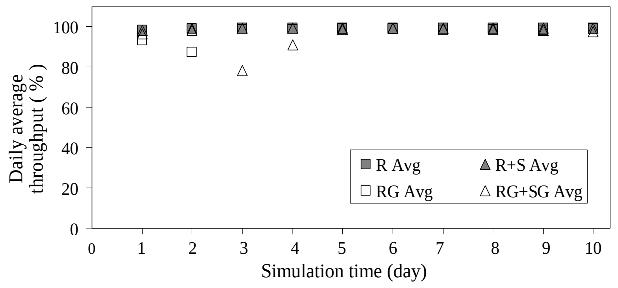

Figure 41 Daily Average Throughput

and platforms evolve and successfully adapt their behavior policies to dynamic network conditions.

At Day 10, the daily average of response time is lower in the scenario R+S than the scenario R, and it is also lower in the RG+SG scenario than the scenario RG. This demonstrates that symbiotic behaviors allow agents and platforms to make themselves available closer to the user and better distribute workload, thereby retaining lower response time.

Fig. 38 shows how resource efficiency changes dynamically in the scenario RG and RG+SG. Fig. 39 shows the daily average of resource efficiency in the scenarios R, R+S, RG and RG+SG. At the beginning of a simulation, the daily average resource efficiency fluctuates in the RG and RG+SG scenarios because agents and platforms still search appropriate behavior policies by altering their genes through evolution. However, agents and platforms evolve and gradually adapt their behavior policies to dynamic network conditions. As a result, the RG and RG+SG results become more stable and closer to the R and R+G results.

At Day 10, the daily average of resource efficiency is higher in the scenario R+S than the scenario R. This demonstrates that agents and platforms can better manage resource efficiency. With symbiotic behaviors, agents can process more service requests and platforms can reduce their availability on unnecessary hosts (e.g., the ones far from the user).

As Figs. 32 to 39 show, similar to manually-configured agents/platforms, evolutionary agents/platforms take advantage of symbiotic behaviors to augment their adaptability (i.e., service adaptation and resource adaptation). Agents and platforms successfully coevolve by finding appropriate behavior policies for symbiotic behaviors.

Fig. 40 shows how throughput changes dynamically in the scenario RG and RG+SG. Fig. 41 shows the daily average throughput in the scenarios R, R+S, RG and RG+SG. At the beginning of a simulation, the daily average throughput fluctuates in the RG and RG+SG scenarios because agents and platforms still search appropriate behavior policies by altering their genes through evolution. However, as the time goes, the RG and RG+SG results become more stable and closer to the R and R+G results. This demonstrates that agents and platforms evolve and autonomously adapt their behavior policies to dynamic network conditions.

In order to compare evolutionary agents/platforms with manually-configured agents/platforms, Fig. 42 shows the performance ratio between evolutionary and manually-configured agents/platforms. Similar to Eq. (6), performance ratio is measured with Eq. (7) with seven performance metrics (response time, throughput, LBI, resource efficiency, platform health level, agent energy level and platform energy level). PG_i denotes the performance in the metric i when agents and platforms obtain their behavior policies through evolution. P_i denotes the performance in the metric i when agents and platforms use manually-configured behavior policies.

$$Performance\ \ Ratio = \sum_{i=1}^{7}\left(\frac{PGi\ -\ Pi}{Pi}\right) \qquad (7)$$

In Fig. 42, each gray bar shows the performance ratio that compares the scenarios R and RG. Each black bar shows the performance ratio that compares the R+S and RG+SG scenarios. At the beginning of a simulation, the performance of evolutionary agents/platforms is worse than that of manually-configured ones. They do not behave adaptively yet because they still search appropriate behavior policies through evolution. As the evolution progresses, evolutionary agents/platforms outperform manually-configured ones. At Day 10, the performance of evolutionary agents/platforms in the RG scenario is 12% better than that of manually-configured ones in the R scenario. Also, evolutionary agents/platforms in the RG+SG scenario perform 63% better than manually-configured ones in the R+S scenario. This result demonstrates that agents and platforms can successfully improve the quality of their behavior policies by themselves. In fact, evolution produces higher quality of behavior policies than the ones that the authors manually configured through trial and errors for 340 hours. In the SymbioticSphere simulator, it takes six hours to run a 10 days simulation. Thus, via offline simulation, a quality set of behavior policies can be obtained though evolution in much shorter time than trial and errors.

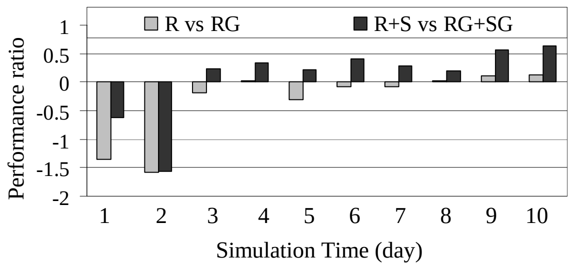

Figure 42. Performance Ratio

Figs. 43 and 44 show the standard deviations of energy utility and behavior invocation efficiency in the agent population, respectively. Note that energy utility and behavior invocation efficiency are two objectives to rank agents. (See Section III.F.) Figs. 43 and 44 demonstrates that the standard deviation of each objective value decreases over time in the agent population. This means that most agents gain appropriate behavior policies through evolution and behave well (adaptively). Agents successfully evolve as a population as well as individuals.

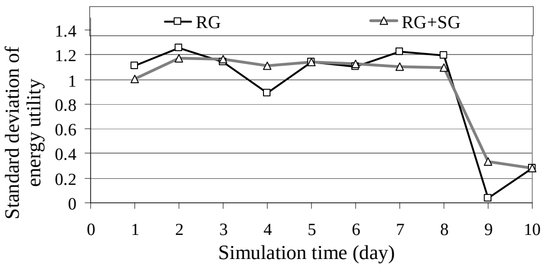

Figure 43. Standard Deviation of Energy Utility

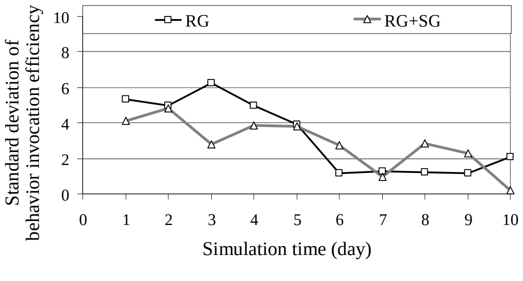

Figure 44. Standard Deviation of Behavior Invocation Efficiency

V. Related Work

This work is an extension to the authors' prior work [17, 18]. [17] shows that agents and platforms improve their adaptability with their regular behaviors; however, the work did not investigate symbiotic behaviors and evolutionary adaptation. [18] shows that symbiotic behaviors improve the adaptability of agents and platforms; however, the work did not study the evolution of agents/platforms. To the best of the authors' knowledge, this paper is the first attempt to investigate how application services (agents) and middleware (platforms) can evolve and coevolve across network layers in order to adapt to dynamic network conditions in a decentralized manner.

[9, 19] propose biologically-inspired agents to achieve service adaptation in a decentralized manner. However, [9, 17] do not consider resource adaptation because platforms are static and non-biological entities. In SymbioticSphere, both agents and platforms are designed as biological entities, and they achieve service adaptation and resource adaptation simultaneously. In addition, SymbioticSphere considers symbiosis between agents and platforms to augment their adaptability.

[19] studies an evolutionary adaptation mechanism in agents. However, the threshold values of behavior policies are not included in genes. This means that agent designers need to manually configure them through trial and errors. In contrast, no manual work is necessary to configure thresholds in SymbioticSphere because they are included in genes. Also, [19] uses a fitness function to rank agents in mating partner selection. It has a weight value for each objective. Agent designers need to manually configure these weight values as well. In SymbioticSphere, no parameters exist for ranking agents/platforms because of a domination ranking mechanism. As a result, SymbioticSphere incurs much less configuration cost.

[20] is designed after population ecology and intended to adapt agent availability to the availability of hosts in a decentralized way. Agent behaviors are governed with the concept of food, which is similar to energy in SymbioticSphere. While [20] follows a single species population model, SymbioticSphere considers two species: agents and platforms. The two species coevolve to augment their adaptability. Unlike [20], which focuses only on the adaptation of agent availability, SymbioticSphere exhibits many other types of adaptation such as the adaptation of response time, throughput, resource availability, resource efficiency and workload distribution.

Rainbow achieves both service adaptation and resource adaptation for grid computing [21]. A centralized server periodically monitors the current network conditions and performs an adaptation strategy such as service migration and platform replication/death. SymbioticSphere provides a wider range of adaptation strategies: more agent/platform regular behaviors (e.g., agent replication and death) and symbiotic behaviors. In SymbioticSphere, agents/platforms perform their behaviors in a decentralized manner. In addition, agents and platforms evolve to adjust their behavior policies even for unanticipated network conditions. However, Rainbow predefines static adaptation strategies for anticipated network conditions; they do not work for unanticipated conditions.

[22] proposes a decentralized design for adaptive data centers that guarantee response time. SymbioticSphere does not guarantee any performance measures because performance improvement (i.e., adaptation) is an emergent and evolutionary product of collective behavior invocations and interactions of agents/platforms. As a result, agents and platforms can adapt to unanticipated network conditions. [22] does not consider to adapt to unanticipated conditions. Unlike [22], which focuses only on response time, SymbioticSphere exhibits many other types of adaptation, such as the adaptation of response time, throughput, resource availability, resource efficiency and workload distribution.

[23] proposes to a centralized evolutionary mechanism to adapt network topology to a given QoS requirement (end-to-end delay of packet transmission). It uses a fitness function to rank genes in its evolution process. The function has a weight value for each objective. Network designers need to manually configure these weight values. SymbioticSphere requires no manual parameter configuration because of its domination ranking. [23] guarantees a required QoS; however, SymbioticSphere does not. It focuses on satisfying service adaptation and resource adaptation simultaneously through coevolution.

[24] implements the concept of symbiosis between different groups of peers (hosts) in peer-to-peer networks. Peer groups symbiotically connect or disconnect with each other to improve search speed and quality. A special type of peers, cooperative peers, implement the symbiotic behaviors and invoke them with fixed policies. In SymbioticSphere, all agents and platforms invoke symbiotic behaviors. They coevolve to dynamically adapt their behavior policies to unanticipated network conditions.

VI. Conclusion

This paper describes the biologically-inspired mechanisms in SymbioticSphere, such as evolution and coevolution as well as regular/symbiotic behaviors, and evaluates their impacts on the adaptability of data centers. Simulation results show that agents and platforms evolve and autonomously adapt to dynamic network conditions Simulation results also show that agents and platforms augment their adaptability through coevolution.

Several extensions to SymbioticSphere are planed. For example, multiple types of agents will be deployed to implement different functional services. (Currently, SymbioticSphere considers a single type of agents.) In order to stabilize energy flows in SymbioticSphere, a certain mechanism will be investigated to assign appropriate service price (in energy units) for each agent type.

REFERENCES

[1] J. Rolia and S. Singhal and R. Friedrich, "Adaptive Internet Data Centers," *Proc. of Int'l Conference on Advances in Infrastructure for Electronics Business, Science, and Education on the Internet,* July 2000.

[2] R. Sterritt and D. Bustard, "Towards an Autonomic Computing Environment," *Proc. of IEEE Int'l Workshop on Database and Expert Systems Applications,* Sept. 2003.

[3] R. Albert, H. Jeong and A. Barabasi, "Error and Attack Tolerance of Complex Networks," *Nature* 406, July 2000.

[4] N. Minar, K. H. Kramer and P. Maes, "Cooperating Mobile Agents for Dynamic Network Routing," *Software Agents for Future Comm. Sys.*, Chap. 12, Springer, 1999.

[5] R. M. Alexander, *Energy for Animal Life,* Oxford University Press, May 1999.

[6] E. Mayr, *What Evolution Is,* Basic Books, 2001.

[7] L. Margulis, *Symbiotic Planet: A New Look at Evolution,* Basic Books, 1998.

[8] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms,* John Wiley & Son, 2001.

[9] J. Suzuki and T. Suda, "A Middleware Platform for a Biologically-inspired Network Architecture Supporting Autonomous and Adaptive Applications," *IEEE J. on Selected Areas in Communications,* 23(2), 2005.

[10] P. Gunaratne, K. Christensen, and B. Nordman, "Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP connections, and Scaling of Link Speed," *Int'l Net. Mgt. J.,* 15(5), 2005.

[11] Advanced Micro Devices, Inc., *Magic Packet Technology,* Technical White Paper 20213, November 1995.

[12] C. Lefurgy, K. Rajamani, F. Rawson, W. Felter, M. Kistler, and T. Keller, "Energy Management for Commercial Servers," *IEEE Computer,* 36(12), 2003.

[13] L. Miller, "Connectivity Properties of Mesh and Ring/Mesh Networks," Technical Report, National Institute of Standards and Technology, April 2001.

[14] R. Fourer, D. M. Gay, and B.W. Kernighan, *AMPL: A Modeling Language for Mathematical Programming,* Duxbury Press, 2002.

[15] B. A. Murtagh and M. A. Saunders, "MINOS 5.5 User's Guide," Technical Report, Stanford University, July 1998.

[16] J. Devale, "High Performance Robust Computer Systems," *Ph.D. dissertation,* Carnegie Mellon University, 2001

[17] P. Champrasert and J. Suzuki, "SymbioticSphere: A Biologically-Inspired Autonomic Architecture for Self-Adaptive and Self-Healing Server Farms," *Proc. of IEEE Int'l Workshop on Autonomic Comm. Comp.*, June 2006.

[18] P. Champrasert and J. Suzuki, "Exploring Adaptive Data Centers through Cooperative Symbiotic Networking," *Proc. of IEEE Int'l Workshop on Future Trends of Distributed Computing Systems,* March 2007.

[19] T. Nakano and T. Suda, "Self-Organizing Network Services With Evolutionary Adaptation," *IEEE Trans. on Neural Networks*, 16(5), 2005.

[20] T. Suzuki, T. Izumi, F. Ooshita and T. Masuzawa, " Self-Adaptive Mobile Agent Population Control in Dynamic Networks Based on the Single Species Population Model," *IEICE Trans. Info. Syst.,* E90-D(1), 2007.

[21] D. Garlan, S. Cheng, A. Huang, B. Schmerl and P. Steenkiste, "*Rainbow: Architecture-Bases Self Adaptation with Reusable Infrastructure*," *IEEE Computer*, 37(10), 2004.

[22] C. Adam and R. Stadler, "Adaptable Server Clusters with QoS Objectives," *Proc. of IFIP/IEEE Int'l Symposium on Integrated Network Management*, May 2005.

[23] D. Montana, T. Hussain and T. Sexana, "Adaptive Reconfiguration of Data Networks Using Genetic Algorithms," *Proc. of Genetic and Evolutionary Comp. Conf.*, July 2002.

[24] J. Konishi, N. Wakamiya and M. Murata, "Proposal and Evaluation of a Cooperative Mechanism for Pure P2P File Sharing Networks," *Proc. of Int'l Workshop on Bio-Inspired Approaches to Advanced Info. Tech.*, Jan. 2006.

Mr. Paskorn Champrasert was born in Bangkok, Thailand in 1976. He is a Ph.D. student at University of Massachusetts, Boston. He received his B.Eng. in computer engineering and M.S. in industrial and organization psychology from Chiangmai University, Thailand. He received his M.S. in computer science from University of Massachusetts, Boston in 2006.

Mr. Champrasert worked for the Department of Computer Engineering, Chiangmai University, as a faculty member from 1998 to 2001. In 1999, he received a Thailand Innovation Award from the Thailand National Research Consortium. In 2002, he worked at the National Science and Technology Development Agency Northern Network (NSTDA-NN) as a project manager for strategic planning and R&D for various organizations in the northern area in Thailand. In 2003, he received Thai Government Scholarship for his Ph.D. work.

Dr. Junichi Suzuki received the Ph.D. in computer science from Keio University, Japan, in 2001.

Dr. Suzuki joined the Department of Computer Science, University of Massachusetts, Boston in September 2004, where he is currently an assistant professor. From 2001 to 2004, he was with the School of Information and Computer Science, University of California, Irvine (UCI), as a postdoctoral research fellow. Before joining UCI, he was with Object Management Group Japan, Inc., as the Technical Director. His research interests include autonomous adaptive distributed systems, biologically-inspired (e.g., ecological, genetic, immunological and developmental) software designs, and model-driven software engineering. In these areas, Dr. Suzuki has authored two books, published over 75 refereed papers including five award papers, and served on technical program committees of over 25 conferences including IEEE AINA 2008 and IEEE/ACM/Create-Net/ICST BIONETICS 2007.

Dr. Suzuki is an active participant and contributor of the International Standard Organization SC7/WG19 and the Object Management Group, Super Distributed Objects SIG. He is a member of IEEE and ACM.