A Biologically-Inspired Autonomic Architecture for Self-Healing Data Centers

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

***Abstract*—**This paper describes a biologically-inspired network architecture, called SymbioticSphere, which allows large-scale data centers to autonomously adapt to dynamic environmental changes and survive partial system failures. SymbioticSphere follows certain biological principles such as decentralization, natural selection, emergence and symbiosis to design data centers (application services and middleware platforms). Each application service and middleware platform is modeled as a biological entity, analogous to an individual bee in a bee colony, and implements biological concepts such as energy level, health level, energy exchange, environment sensing, migration, replication and death. Simulation results show that, like in biological systems, desirable system properties in data centers (e.g., adaptability and survivability) emerge from collective actions and interactions of application services and platforms.

1. Introduction

Data centers have become integral components to operate large-scale Internet services. Since they are rapidly increasing in complexity and scale, they face several challenges, particularly *adaptability* and *survivability*. Data centers are expected to autonomously adapt to dynamic environment changes such as demand surges and resource exhaustion [1, 2]. They are also expected to autonomously recover (or self-heal) from partial system failures due to, for example, errors by administrators and physical damage to data center fabric as a result of unexpected events (e.g., hurricanes, blackouts or terrorism acts) [3, 4]. Since the majority of the current data centers are manually configured, the degree of autonomous adaptability and survivability is limited; they often result in poor user experience, lower system availability and higher maintenance cost [1, 5].

Based on the observation that various biological systems have already achieved the above requirements (i.e., autonomy, adaptability and survivability), the proposed architecture, called SymbioticSphere, applies biological principles to design large-scale data centers (application services and middleware platforms). We believe if data centers adopt certain biological principles, they may be able to meet the above requirements.

In SymbioticSphere, each application service and middleware platform is modeled as a biological entity, analogous to an individual bee in a bee colony. Both application services and middleware platforms are designed to follow several biological principles such as decentralization, natural selection, emergence and symbiosis. An application service is implemented as an autonomous software agent. Each agent implements a functional service and follows simple biological behaviors such as replication, death, migration and energy exchange. A middleware platform runs on a network host and operates agents. Each platform provides a set of runtime services that agents use to perform their services and behaviors, and implements simple biological behaviors such as replication, death and energy exchange.

This paper describes the biologically-inspired mechanisms in SymbioticSphere and evaluates their impacts on the adaptability and survivability of data centers. Simulation results show that agents and platforms autonomously adapt to dynamic environment changes and survive (self-heal) partial system failures to retain their availability and performance.

2. Design Principles in SymbioticSphere

SymbioticSphere consists of two components: agents and middleware platforms. Agents run on platforms, which in turn run on network hosts. Agents and platforms are designed based on the following principles.

(1) Decentralization: There are no central entities to control and coordinate agents/platforms (i.e., no directory servers and no resource managers). Decentralization allows agents/platforms to be scalable, survivable and simple by avoiding a single point of performance bottlenecks and failures [6, 7] and by avoiding any central coordination in deploying agents/platforms [8].

(2) Autonomy: Agents and platforms sense their local network environments, and based on the sensed environmental conditions, they autonomously behave, and interact with each other without any intervention from/to other agents, platforms and human users.

(3) Natural selection: Agents and 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 (e.g., memory space and CPU). Each platform gains energy in exchange for providing resources to agents, and continuously evaporates energy. The abundance or scarcity of stored energy triggers to execute natural selection of agents/platforms. For example, an abundance of stored energy indicates higher demand for an agent/platform; thus the agent/platform replicates itself. A scarcity of stored energy (an indication of lack of demand) causes death of the agent/platform.

Like in biological natural selection where more favorable species in a particular environment becomes more abundant, the population of agents/platforms dynamically changes based on the demands for them.

(4) Emergence: Agents and platforms behave against dynamically changing environment conditions (e.g., user demands, user locations and resource availability). For example, an agent may invoke migration behavior to move towards a platform that forwards a large number of request messages for its services. Also, a platform may replicate itself on a neighboring host where resource availability is high. SymbioticSphere is designed to exhibit emergence of desirable system properties such as adaptability and survivability. In a swarm of agents and platforms, these properties emerge from collective behaviors and interactions of individual agents and platforms, although the properties are not present in any single agent/platform.

(5) Symbiosis: SymbioticSphere models agents and platforms as different species. In certain circumstances, agents and platforms spontaneously cooperate in a symbiotic manner to pursue their mutual benefits (i.e., to increase their adaptability and survivability), although each of them is not explicitly designed to do so.

3. SymbioticSphere

This section presents the design of SymbioticSphere.

3.1 The Architecture of SymbioticSphere

SymbioticSphere models agents and platforms as different species, and follows ecological principles to design energy exchange among agents, platforms and environment. Fig. 1 shows a simplified energy flow in the ecological system. The Sun gives light energy, and producers (e.g., plants and microorganisms) convert it to chemical energy. The chemical energy flows through multiple species, called consumers. It will be eventually transferred to decomposers (e.g., bacteria and fungi). For example, shrubs (producers) convert the Sun light energy to chemical energy, hares (primary consumers) consume shrubs, and foxes (secondary consumers) consume hares. In energy exchange between different species, it is known that about 10% of the energy maintained by one species is transferred to another species [9]. The remaining 90% of the energy is used for metabolism, growth and actions/behaviors (e.g., moving and reproduction).

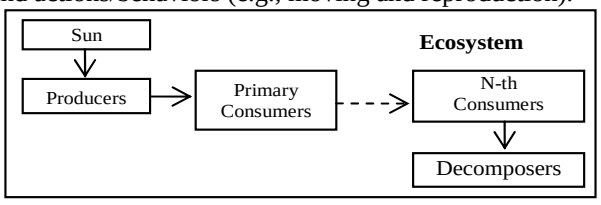

Fig. 1 Energy Flow in Ecosystem

Fig. 2 shows the energy exchange in SymbioticSphere. SymbioticSphere models each user as the Sun, agents as producers and platforms as (primary) consumers. Similar to the Sun, users have unlimited amount of energy. Agents gain energy from users[1], and expend energy to consume resources provided by platforms (e.g., memory space). They periodically transfer 10% of the current energy level to platforms on which they operate. Platforms periodically gain energy from agents, and evaporate 10 % of the current energy level to the environment.

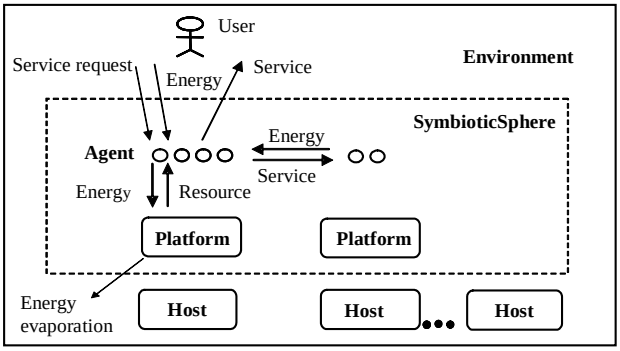

Fig. 2 Energy Exchange in SymbioticSphere

3.2 Agents

Each agent consists of three parts: *attributes*, *body* and *behaviors*. *Attributes* carry descriptive information regarding the agent, such as agent ID, energy level and description of a service it provides. *Body* implements a service that the agent provides. For example, an agent may implement a web service and contains web pages in its body while another agent may implement a physical model for scientific simulations. *Behaviors* implement actions that are inherent to all agents. Although SymbioticSphere defines nine standard agent behaviors [10], this paper focuses on three of them.

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

3.3 Platforms

Each platform runs on a network host and operates agents. It consists of *attributes*, *behaviors* and *runtime services*. *Attributes* carry descriptive information regarding the platform, such as platform ID, energy level and health level. *Health level* indicates how healthy an underlying host is. It is defined as a function of resource availability on, age of and freshness of a host. Resource

[1] Each agent specifies the price (in energy units) of service that it provides.

availability indicates how much resources (e.g. memory space) are available for agents and platforms on a host. Age indicates how long a host has been alive (i.e. how much stable a host is). Freshness indicates how recently a host joined the network. After a new host joined the network, its freshness gradually decreases from a maximum value. When a host resumes from a failure, its freshness starts with the value that the host has when it went down. Using age and freshness, unstable hosts and new hosts can be distinguished. Unstable node tends to have lower freshness and higher age, and new hosts tend to have higher freshness and lower age (Table 1).

Table 1.Freshness and Age in Different Types of Hosts

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

Health level affects behaviors of a platform and agent. For example, higher health level indicates higher stability of and higher resource availability on a host that a platform resides on. Thus, the platform may replicate itself on a healthier neighboring host.

Behaviors are the actions inherent to all platforms.

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

Runtime services are middleware services that agents and platforms use to perform their behaviors. In order to maximize decentralization and autonomy of agents/platforms, they only use their local runtime services. They are not allowed to invoke any runtime services running on a remote platform.

3.4 Behavior Policies of Agents and Platforms

Each agent and platform has policies for its behaviors. A behavior policy defines when to and how to invoke a particular behavior. Each behavior policy consists of one or more *factors* (F_i), which evaluate environment conditions (e.g. network traffic and resource availability) or the status of agent/platform/host (e.g. energy level and health level)[2]. Each factor is given a *weight* (W_i) relative to its importance. Behaviors are invoked if the weighted sum of factor values ($\Sigma F_i * W_i$) exceeds a threshold.

The factors in agent migration behavior include:

- *Health level Ratio*: encourages agents to move to platforms running on healthier hosts. Health level ratio is calculated with the following equation. HostProperty includes resource availability, freshness and age.

$$HealthLevelRatio = \sum_{i}^{N}\left(\frac{RemoteHostProperty_i - LocalHostProperty_i}{LocalHostProperty_i}\right) \quad \text{.....(1)}$$

- *Service Request Ratio*: the ratio of the number of service requests on a remote platform by the number of service requests on a local platform, which encourages agents to move towards users.
- *Migration interval*: interval from the time of a previous migration, which discourages agents to migrate too often.

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

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

Platform replication behavior has a factor of health level ratio, which encourages platforms to replicate themselves on a healthier neighboring host. A replicated (child) platform is placed on a host whose health level is highest among neighboring hosts.

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

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

4. Simulation Results

This section shows simulation results to evaluate how biologically-inspired mechanisms in SymbioticSphere impact the self-healing (survivability) of data centers[3]. In this paper, self-healing means the ability of agents and platforms to autonomously survive unexpected events (e.g., host failures, network link failures and demand surges) through their adaptation activities and to retain the availability (service availability and resource availability) and performance (response time and throughput) of data centers. Service availability is measured as the number of available agents. Resource availability is measured as the number of platforms that make resources available for agents.

Fig. 3 shows a pseudo code to run users, agents and platforms in each simulation cycle.

[2] Each agent and platform can sense its local environment. An agent can sense agent population, network traffic and resource availability on a local and neighboring platforms. A platform can sense agent population on itself, and health level of a local and neighboring hosts.

[3] Simulations were carried out with the SymbioticSphere simulator, which contains 14,120 lines of Java code. It is available for researchers who investigate autonomic network systems (dssg.cs.umb.edu).

```
While (not the last cycle in a simulation)
    For each user Do
        Send service requests to each of available agents according to
         a configured service request rate.
    End For
    For each platform Do
        Make a decision on replication and death behaviors.
        Update health level.
        Expend (evaporate) energy.
    End For
    For each agent Do
        If (a service request(s) arrived)
            Process the request(s) and gain energy.
        End If
        Make a decision on replication, migration and death behaviors.
        Expend energy to the local platform.
    End For
End While
```

Fig. 3 Pseudo Code of Each Simulation Cycle

When a user issues a service request, the service request is passed to the local platform where the user resides on, and the platform performs a discovery process to search a target agent that can process the issued service request. The platform (discovery originator) forwards a discovery message to its neighboring platforms, asking whether they host a target agent. If a neighboring platform hosts a target agent, it returns a discovery response to the discovery originator. Otherwise, it forwards the discovery message again to its neighboring platforms. Fig. 4 shows this peer-to-peer agent discovery through platform connectivity[4].

```
While (not simulation last cycle)
   If ( Discovery messages arrived)
      For each of discovery messages (under the max # of messages to be
            processed in each simulation cycle) Do
            If ( Discovery message matches one of the local agents)
                Returns a discovery response to discovery originator
            Else
                Forward the discovery message to neighboring platforms
            End If
      End For
   End If
End While
```

Fig. 4 Pseudo Code for Agent Discovery Process in each Simulation Cycle

Fig. 4 shows a simulated network. A data center consists of hosts connected in an *N* x *N* grid topology, and service requests travel from users to agents via user access point. This simulation study assumes that a single (emulated) user runs on the access point and sends service requests to agents. Each host has 256MB memory[5]. Out of the memory space, an operating system consumes

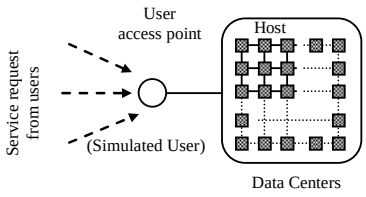

Fig. 4 Simulated Network

[4] Note that there is no centralized directory to keep track of agents.

128 MB, and Java virtual machine consumes 64MB. Thus, 64MB is available for a platform and agents on each host. Each agent and platform consumes 5 MB and 20 MB, respectively. This assumption is obtained from a prior empirical experiment [10].

4.1. Self-Healing against Demand Surges

The first simulation study evaluates the self-healing of agents and platforms to survive unexpected demand surges. A simulation runs for 24 hours in simulation time. Service request rate starts with 3,000 requests/min, spikes to 210,000 requests/min at 8:00, and drops to 3,000 requests/min at 16:30 (Fig. 6). The peak demand and spike ratio (1:70) are taken from a workload trace of the 1998 World Cup web site [11]. A simulated data center is 7x7 (49 hosts) from 0:00 to 12:00 and 15x15 (225 hosts) from 12:00 to 24:00. At the beginning of simulation, an agent and platform is deployed on each host.

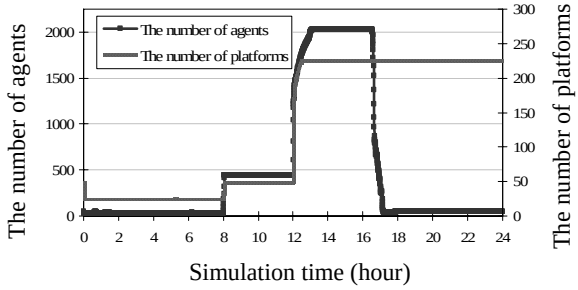

Fig. 5 The Number of Agents and Platforms

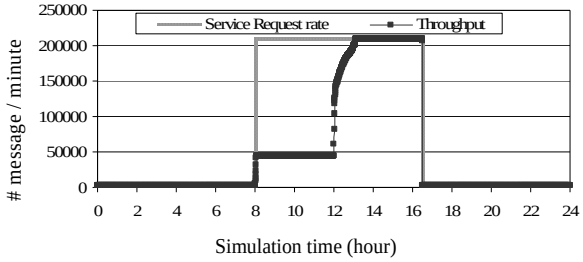

Fig. 6 Service Request and Throughput

Fig. 5 shows how service availability (i.e., the number of agents) and resource availability (i.e., the number of platforms) change dynamically. Starting with 49 agents and 49 platforms at 0:00, they autonomously adapt their population to demand changes. When service request rate spikes at 8:00, agents gain more energy form users 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 replications. From 8:00 to 12:00, the populations of agents and platforms do not grow due to physical limitation of available hosts. When the size of a data center expands from 49 hosts to 225 hosts at 12:00,

[5] Currently, memory availability represents resource availability on each platform/host.

agents and platforms recognize the environment change and rapidly perform replications and migrations on the new hosts. When service request rate drops at 16:30, most agents die due to energy starvation because they cannot balance energy gain and expenditure. The population of platforms also decreases due to less energy transfer from agents. (It happens after 24:00.) Fig. 5 shows that biological mechanisms in SymbioticSphere contribute for agents and platforms to adaptively adjust their availability to dynamic demand changes, and avoid their crashes or malfunctions due to overloading.

Fig. 6 shows the service request rate from users and the throughput achieved by agents. At 8:00, service request rate spikes. Through replications, agents increase throughput to 5,000 messages/min. The throughput stops increasing because of the limitation of available hosts for agents to utilize. When new 176 hosts are introduced to a data center at 12:00, platforms replicate themselves on the new hosts, and agents increase throughput by migrating to the replicated platforms and replicating themselves on the platforms. Fig. 6 shows that the biological mechanisms in SymbioticSphere contribute for agents and platforms to adapt their availability to dynamic demand changes and host availability, and collectively retain throughput performance.

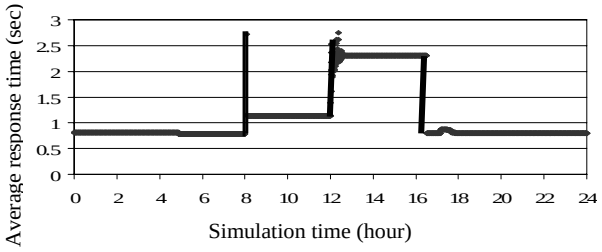

Fig. 7 Average Response Time

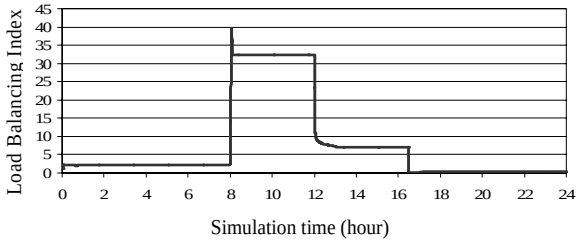

Fig. 8 Load Balancing Index

Fig. 7 shows the average response time for agents to respond users. This includes the message transmission latency between a user and agent, and the processing overhead for an agent to process a service request. From 0:00 to 8:00, response time gradually decrease because agents migrate closer to users. At 8:00, response time spikes because service request rate spikes. Agents increase their population through replications, and their response time drops from 2.7 to 1.1 seconds at 8:20. When new hosts are added to a data center at 12:00, agents increase their population again and some of them migrate to the newly added hosts, in order to increase their throughput. As a result, the average distance between agents and users increases, and agent response time also increases. However, agents keep response time low enough (2.3 seconds) while they achieve the throughput of 210,000 requests/min. Fig. 7 shows the biological mechanisms in SymbioticSphere contribute for agents and platforms to recover the degradation of their response against service demand surge.

Fig 8 shows dynamic changes in Load Balancing Index (LBI), which indicates how workload is distributed over available platforms. (the lower, the better.) LBI is measured with the following equation.

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

X_i represents (the # of messages processed by agents running on platform i) / (resource utilization on platform i). μ represents the expected average of X, which means (the total # of messages processed by all agents) / (the total amount of resource utilization on all platforms) / (the number of platforms; N). LBI drops at 12:00, because platforms replicate themselves on newly available hosts, and agents migrate to and perform replications on the replicated platforms. Fig. 8 shows that biological mechanisms in SymbioticSphere contribute for agents and platforms to collectively avoid overloading platforms by balancing workload over them.

Fig. 8 also shows an example of symbiotic emergence between agents and platforms. Agent migration behavior policy encourages agents to move towards platforms on healthier hosts. Platform replication behavior policy encourages platforms to replicate themselves on healthier hosts. As a result, workload is spread over the platforms running on healthy hosts. This contributes to balance workload on each platform, although agent migration policy and platform replication policy do not consider agent population, platform population and load balancing. This results in a mutual benefit for both agents and platforms. Platforms help agents increase their throughput by making more resources available for them. Agents help platforms to keep their stability by avoiding excessive resource utilization on them.

4.2. Self-Healing against Host Failures

The next simulations study evaluates the self-healing of agents and platforms against host failures. Each simulation runs for 12 hours in simulation time. Service request rate is constantly 7,200 requests/min, which was the peak in a workload trace of the IBM web site in 2001 [12]. In each simulation, randomly chosen 30% or 60% of hosts go down at 5:00 for an hour. The size of a data center is 7x7 (49 hosts). At the beginning of simulation, an agent and platform is deployed on each host.

Figs. 9 and 10 show resource availability (i.e., the number of platforms) and service availability (i.e., the number of agents), respectively. Fig. 11 shows the throughput of agents. Fig. 12 shows the average response

time of agents in the case that 60% of hosts go down. When a host goes down, agents and platforms running on the host die. This is why throughput drops from 7,200 requests/min and response time spikes from 1 second at 5:00. After host failures, the remaining agents increase their populations through replications on remaining hosts (Fig. 10), and throughput goes back to 7,200 requests/min (Fig. 11). When failed hosts are back to a data center, platforms recognize that and replicate themselves on the hosts (Fig. 9). Some of agents migrate to the replicated platforms in order to come near users. As a result, response time drops back to 1 second (Fig. 12). Figs. 9, 10, 11 and 12 show that the biological mechanisms in SymbioticSphere contribute for agents and platforms to survive host failures and retain service availability, resource availability, throughput performance and response time performance.

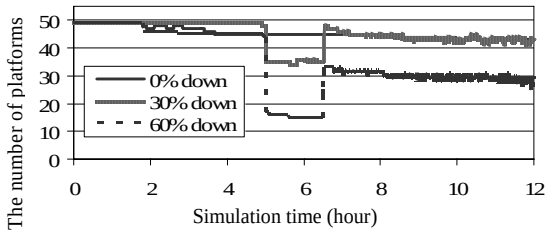

Fig. 9 The Number of Platforms

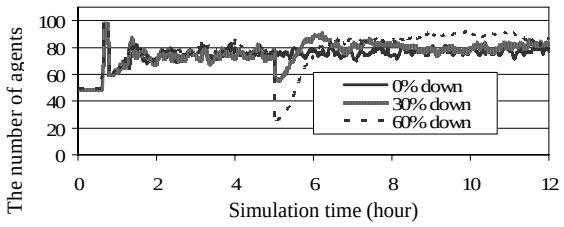

Fig. 10 The Number of Agents

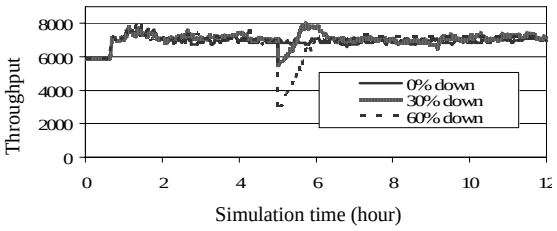

Fig. 11 Throughput

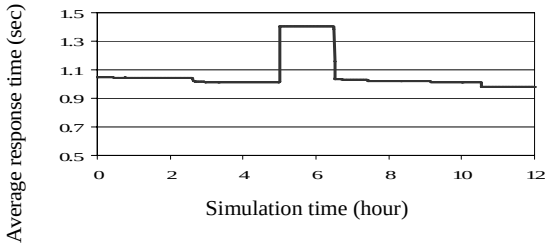

Fig. 12 Average Response Time in the case that 60% of hosts fail

4.3. Self-Healing against Data Center Failures

The next simulation study evaluates the self-healing of agents and platforms against data center failures. Each simulation runs for 24 hours in simulation time. Service request rate is constantly 7,200 requests/min. At the beginning of a simulation, an agent and a platform are deployed in each host. In this simulation study, there are two data centers as shown in Fig. 13. Service requests are distributed to the two data centers evenly (i.e., in a round-robin manner). The size of each data center is 5x5 (25 hosts). Data center 1 and data center 2 go down for 4 hours at 6:00 and 18:00, respectively, due to network link failures between a data center and the user access point. When a failed network link is fixed, this simulation study assumes that an administrator deploy an agent and a platform on each host again.

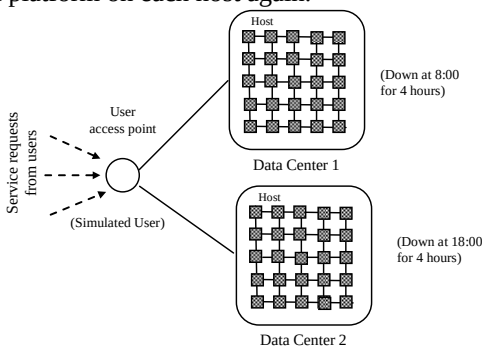

Fig. 13 2 Data Centers

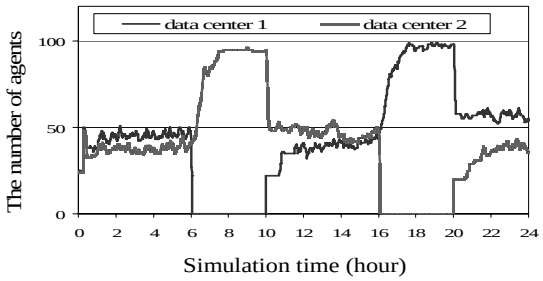

Fig 14 The Number of Agents in Data Center 1 and 2

Fig. 14 shows how service availability (i.e., the number of agents) changes in data centers. When a network link goes down between the user access point and data center 1 at 6:00, agents and platforms in the data center die off in about 15 minutes due to energy starvation. In this failure, all service requests are forwarded to data center 2. (the workload in data center 2 changes from 3,600 requests/min to 7,200 requests/min.) In response to demand changes, agents increase their populations through replications. At 10:00, a link failure is fixed and data center 1 becomes available again. Since service requests are distributed to data centers 1 and 2, agents in data center 1 increase their populations through replications, and agents in data center 2 decrease their populations through death. As a result, the number of agents in the two data centers becomes close with each other. At

16:00, data center 2 becomes unavailable due to a network link failure between the user access point and data center 2. Agents and platforms in data center 2 die off in a short time, and agents in data center 1 increase their populations through replications in response to demand changes in the data center. Fig. 14 shows that the biological mechanisms in SymbioticSphere contribute for agents to survive data center failures (or network link failures) and retain service availability for users.

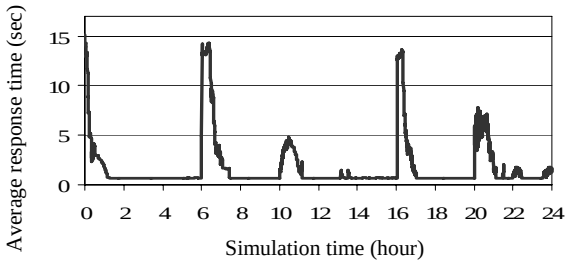

Fig 15 Average Response Time

Fig. 15 shows the average response time of agents. From 0:00 to 1:00, agents reduce their response time through replicating themselves and migrating towards users. From 1:00 to 6:00, response time is constantly at 0.62 second. When data center 1 becomes unavailable at 6:00, response time spikes up to 15 seconds. Then, agents in data center 2 increases their population to compensate of the failure of data center 1, and decrease average response time to 0.65 second at 8:00. When data center 1 becomes available again at 10:00, response time increases while agents in data center 1 replicate themselves to efficiently respond users. At 11:00, response time becomes stable at 0.62 second. When data center 2 becomes unavailable at 16:00, agents in data center 1 increases their population to compensate of the failure of data center 2, and decrease average response time. Fig. 15 shows the biological mechanisms in SymbioticSphere contributes for agents to survive data center failures (network link failures) and retain response time for users.

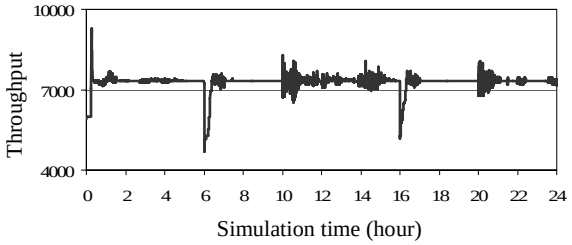

Fig 16 Throughput

Fig. 16 shows the throughput of agents. Throughput drops when data centers become unavailable at 6:00 and 16:00, because agents in a failed data center do not respond users. However, remaining agents in a remaining data center cover the failure and increase throughput to process 7,200 requests/min in a short time. Fig. 16 shows that the biological mechanisms in SymbioticSphere contribute for agents to survive data center failures (network link failures) and maintain high throughput for users.

5. Related Work

This work is an extension to the Bio-Networking Architecture [10, 13]. In this architecture, biologically-inspired agents achieve adapt their population and locations to dynamic environment changes in a decentralized manner. However, platforms do not adapt to environment changes because they are static and non-biological entities. Also, the Bio-Networking Architecture does not study survivability (self-healing) of agents and platforms. In SymbioticSphere, both agents and platforms are biological entities, and they achieve autonomous adaptability and survivability. SymbioticSphere also exhibits a new from of adaptation, symbiotic emergence, which does not appear in the Bio-Networking Architecture.

Resource Broker [14] and Muse [12] are designed to dynamically adjust resource allocation for server clusters via centralized system monitor. Resource Broker inspects the stability and resource availability of each host, and adjusts resource allocation for applications. Muse inspects electric power consumption by a server cluster and adjusts resource allocation for applications. Rather than following centralized architectures, SymbioticSphere achieves adaptive allocation of agents and platforms in a decentralized manner. Resource Broker and Muse do not consider survivability (self-healing) against system failures.

The Willow Architecture [15] addresses survivability of distributed systems. Centralized system components monitor and analyze the operating environment conditions and detect failures. Based on type of failures, Willow performs system reconfiguration to, for example, add, removes and replace system components. It tends to be complicated to define and operate system reconfiguration in Willow. In contrast, SymbioticSphere does not require any complex survivability configuration before running agents and platforms. It also does not depend on any centralized system components to avoid single point of failures.

The concept of energy in SymbioticSphere is similar to money in economy. MarketNet [16] and WALRAS [17] apply the concept of money to address market-based access control for network applications. Instead of access control, SymbioticSphere focuses on adaptability and survivability of network systems (network applications and middleware platforms).

[18] implements the concept of symbiosis between groups of peers 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 implements the symbiotic behaviors in peer groups. Since the number of them is statically fixed, they do not scale well to network size and traffic. They also do not address survivability of peers from host failures. In

SymbioticSphere, all agents and platforms can spontaneously exhibit symbiotic behaviors. They scale well to network size and traffic, and survive from host failures.

[19] implements the programming paradigm based on the actions of biological cells and demonstrates the ability of systems to survive massive failures. However, the robustness of the demonstrated network is achieved because of configuration parameters (the number of transmission attempts and time out). In SymbioticSphere, service availability (the number of agents) and resource availability (the number of platform) are adapted automatically to the environment conditions. SymbioticSphere scale well to network size and traffic, and survive from host failures.

Termite [20] is a biologically inspired algorithm to route messages in mobile wireless ad-hoc networks. Termite applied the concept of four social insects principles (positive feed back, negative feed back, randomness, and multiple interactions). The routing table in each node is locally updated on the run time based on the probabilistic function and historical data. The termite can achieve robustness of routing through the use of multiple paths. However, the initial values of configuration parameters are fixed and randomly generate. The authors cannot provide the relationship between configuration parameters and optimal results. SymbioticSphere focuses on the wired network. Robustness is achieved by symbiosis of agent and platform based on the current environment condition.

6. Concluding Remarks

This paper overviews SymbioticSphere, and presents how it implements biological principles to improve adaptability and survivability (self-healing) of large-scale data centers. Simulation results show SymbioticSphere allows data centers to autonomously adapt to and survive partial system failures.

Reference

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

[2] J. Rolia and S. Singhal and R. Friedrich, “Adaptive Internet Data Centers,” *Proc. SSGRR’00*, July 2000.

[3] Nguyen-Tuong, A. S. Grimshaw, G. Wasson, M. Humphrey, J.C. Knight, “Towards Dependable Grids,” University of Virginia, TR-CS-2004-11, 2004.

[4] A Fox and D Patteron, “Self-Repairing Computers,” *Scientific American*, June 2003.

[5] S. Ranjan and J. Rolia and E. Knightly and H. Fu, “QoS-Driven Server Migration for Internet Data Centers,” *Proc. of IWQoS*, 2002.

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

[7] N. Minar, K. H. Kramer and P. Maes, “Cooperating Mobile Agents for Dynamic Network Routing,” *Software Agents for Future Communications Systems*, Chapter 12, Springer, 1999.

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

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

[10] 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 Comm.* February 2005.

[11] M.F. Arlitt and T. Jin, “A Workload Characterization Study of the 1998 World Cup Web Site,” *IEEE Network*, vol. 14, no. 3, pp. 30–37, May/June 2000

[12] J. Chase, D. Anderson, P. Thakar, and A. Vahdat, “Managing Energy and Server Resources in Hosting Centers,” *Proc. of ACM SOSP'01*, October 2001

[13]

[14] 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,” *The Internet as a Large-Scale Complex System*, Oxford University Press, June 2005.

[15] A. Othman, P. Dew, K. Djemame, I, Gourlay, “Adaptive Grid Resource Brokering,” *Proc. of IEEE Int’l Conference on Cluster Computing*, Dec. 2003

[16] J. C. Knight, D. Heimbigner, A. Wolf, A. Carzaniga, J. Hill, P. Devanbu, and M. Gertz, " The Willow Architecture: Comprehensive Survivability for Large-Scale Distributed Applications," In *Proc. of the International Conference on Dependable Systems and Networks (DSN-2002), June 2002.*

[17] Y. Yemini, A. Dailianas, and D. Florissi, ”MarketNet: A Market-based Architecture for Survivable Large-scale Information Systems," *Proc. of ISSAT International Conference on Reliability and Quality in Design*, Aug. 1998

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

[19] N. Wakamiya and M. Murata, ``Toward Overlay Network Symbiosis," *Proc. of P2P 2005*, September 2005.

[20] S. George, D. Evans, and S. Marchette, “A Biologically Inspired Programming Model for Self-healing System,” First ACM Workshop on Survivable and Self-Regenerative Systems, October 31, 2003

[21] M. Roth, S. Wicker, “Termite: Ad-Hoc Networking with Stigmergy,” In Proc. of Globecom 2003, December 2003.