SymbioticSphere: a Network Architecture for Autonomic and Emergent Networking

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

***Abstract*—Network systems are expected to be more scalable and more adaptive to dynamic network environments. Based on the observation that various biological systems have already overcome these requirements, the proposed network architecture, called SymbioticSphere, applies biological concepts and mechanisms to design network systems (i.e. application services and middleware platforms). In SymbioticSphere, each application service and middleware platform is designed as artificial biological entity, analogous to an individual bee in a bee colony. Application services and middleware platforms implement biological concepts and mechanisms such as decentralization, energy level, health level, energy exchange, environment sensing, migration, replication and death. Like in biological systems, desirable system characteristics such as scalability and adaptability emerge from collective actions and interactions of application services and platforms.**

I. INTRODUCTION

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

In SymbioticSphere, each application service and middleware platform is modeled as a biological entity, analogous to an individual bee in a bee colony. An application service is designed as an autonomous and distributed 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 (application services). Each platform implements a set of runtime services that agents use to perform their services and behaviors, and follows biological behaviors such as replication, death and energy exchange.

Similar to biological entities, agents and platforms in SymbioticSphere store and expend *energy* for living. Each agent gains energy in exchange for performing its service to other agents or human users, and expends energy to use network and computing resources. Each platform gains energy in exchange for providing resources to agents, and continuously evaporates energy to the network environments. SymbioticSphere models agents and platforms as different species, and follows several concepts in ecological food chain to determine how much energy agents/platforms expend at a time and how often they expend energy[1]. The abundance or scarcity of stored energy affects behaviors of an agent/platform. For example, an abundance of stored energy indicates higher demand for the agent/platform; thus the agent/platform may be designed to favor replication in response to higher energy level. A scarcity of stored energy (an indication of lack of demand) may cause death of the agent/platform.

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

This paper is organized as follows. Section II summarizes key design principles of SymbioticSphere. Section III describes the designs of agents and platforms in SymbioticSphere. Section IV shows simulation results. Sections V and VI conclude with discussion on related work and future work.

II. DESIGN PRINCIPLES IN SYMBIOTICSPHERE

SymbioticSphere consists of two major components: agents (applications services) and middleware platforms. Agents run on platforms, which in turn run on network hosts. Agents and platforms are designed based on the following principles [4].

(1) Decentralization: Agents and platforms are decentralized. 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 and simple by avoiding a single point of performance bottleneck [5] and by avoiding any central coordination in developing deploying agents/platforms [6].

(2) Autonomy: Agents and platforms are autonomous. They monitor their local network environments, and based on the monitored environmental conditions, they autonomously behave, and interact without any intervention from/to other agents, platforms and human users.

[1] Agents expend more energy more often when receiving more energy from users. Platforms expend more energy more often when receiving more energy from agents. See [3] for details on energy exchange in SymbioticSphere.

(3) Adaptability: Agents and platforms are adaptive to dynamically changing environment conditions (e.g. user demands, user locations and resource availability). Adaptation is achieved through designing agent/platform behavior policies to consider local environment conditions. For example, agents may implement a migration policy of moving towards a platform that forwards a large number of request messages for their services. This results in the adaptation of agent locations, and agents concentrate around the users who request their services. Also, platforms may invoke replication and death behaviors when their energy levels become over and below thresholds. This results in the adaptation of platform population, and platforms adjust resource availability on them against the demands for resources.

III. SYMBIOTICSPHERE

This section describes the designs of agents and platforms.

A. 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 genetic algorithm for an optimization problem, 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 [4], this paper focuses on three of them.

- *Migration:* Agents may move from one platform to another.
- *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 may die due to energy starvation. When an agent dies, an underlying platform removes the agent from the network and releases all resources allocated to the agent.

B. Platforms

Each platform runs on a network host and operates agents[2]. It consists of *attributes*, *behaviors* and *runtime services*.

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

Behaviors are the actions that are inherent to all platforms. Although SymbioticSphere defines six standard platform behaviors [4], this paper focuses on two of them.

- *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 may die due to the lack of energy. A dying platform uninstalls itself from the network and releases all resources the platform uses. Despite the death of a platform, an underlying network host remains active so that other platforms can run on it in the future.

Runtime services are 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.

C. Behavior Policies of Agents and Platforms

Each agent and platform has policies for its behaviors. A behavior policy defines when to and how to invoke a particular behavior. Each behavior policy consists of one or more *factors* (F_i), which evaluate environment conditions (e.g. network traffic) or agent/platform status (e.g. energy level and health level). 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:

- *Service Request Ratio*, (# of service requests on a remote platform)/(# of service requests on a local platform), which encourages agents to move towards users.
- *Health Level Ratio*, (health level of a remote host)/(health level on a local host), which encourages agents to move to platforms running on healthier hosts.
- *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 factor values for each of the platforms, and migrates 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.

The factors in platform replication behavior include:

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

[2] Currently, SymbioticSphere assumes that at most one platform runs on each network 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. Each platform never performs death behavior while an agent(s) runs on the platform.

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

VI. Preliminary Simlation Results

This section shows preliminary simulation results to evaluate how agents and platforms in SymbioticSphere achieve scalability and adaptability[3]. This simulation work assumes agents implement an application (e.g. grid application and Internet data center application) on a wired network.

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

A simulated network is an 8x8 grid topology network with 64 network hosts (Fig. 1). At the beginning of each simulation, a platform is initialized on the host 63, and an agent is deployed on the platform. There are two types of hosts: resource-poor and resource-rich hosts. A resource-poor host has 320MB memory, and a resource-rich host has 512MB memory[4]. On each host, an operating system consumes 128MB, and a Java virtual machine consumes 64MB. Thus, 320MB and 128MB are available for a platform and agents in resource-rich and resource-poor hosts, respectively. Each agent and platform consumes 5MB and 20MB, respectively. This assumption is obtained from a prior empirical experiment [4].

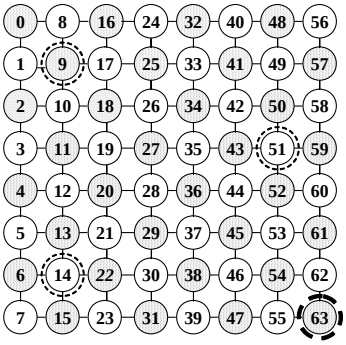

Fig. 1. Simulated network.

This simulation work implements two different networks: homogeneous and heterogeneous networks. In the homogeneous network, all 64 hosts are resource-poor. The heterogeneous network has 32 resource-poor hosts and 32 resource-rich hosts. Shaded circles in Fig. 1 mean resource-rich hosts.

Three users request services provided by agents. Fig. 2 shows how each user changes its service request rate for 24 hours (from 0:00 to 24:00). Two users always reside on hosts 9 and 14, and the third user enters the network (on host 51) at 8:00. Transmission latency of service request is 0.1 second between two hosts. An agent processes an incoming service request in 0.2 second.

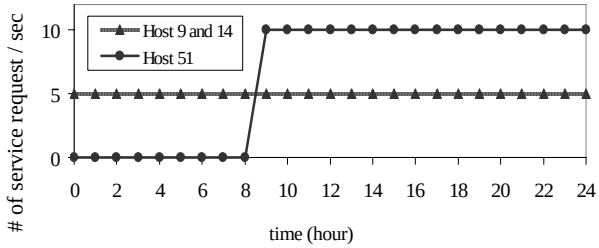

Fig. 2. Service request rate from users.

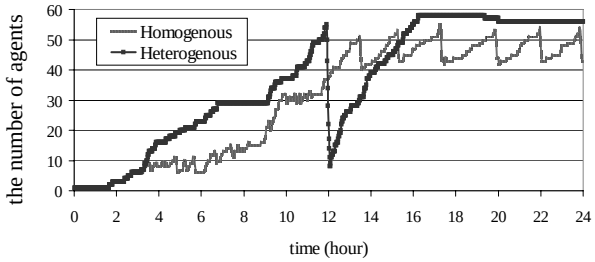

Fig. 3. The number of agents.

Fig. 3 shows how service availability (i.e. the number of agents) changes. Agents adapt their population as a group to the demand for their services. When service request rate spikes from 8:00 to 9:00, agents autonomously increase service availability more rapidly because they gain more energy from users and perform replication more often. This result shows agents scale well to rapid changes in service demand. In the heterogeneous network, agents keep replicating themselves and stay on the host 63 for a while because the health level of the host 63 is much higher than those of its neighboring hosts. At 12:00, some of the agents running on the host 63 migrate to the hosts 55 and 62. The agents process all the service requests from users; no service requests reach the host 63. As a result, agents on the host 63 die off, and this leads to a rapid drop in the number of agents at 12:00. From 12:00 to 16:00, the number of agents increases again. In this period, agents perform replications on the hosts 55 and 62, platforms are replicated on neighboring hosts (the hosts 47 and 54), and agents perform further replications on those hosts.

[3] Simulations were carried out with the SymbioticSphere simulator, which contains 14,100 lines of Java code (dssg.cs.umb.edu/symbiosis/). It is freely available for researchers who investigate autonomic network systems.

[4] Currently, memory availability represents resource availability.

Fig. 4 shows how resource availability (i.e. the number of platforms) changes. Platforms adaptively improve resource availability by changing their population against the demand for their resources. In the homogeneous network, agents rapidly increase their population when the third user enters the network at 8:00. When service request rate becomes high, agents gain more energy from users and transfer more energy to platforms for using their resources. Then, higher energy level contributes platform replication. In heterogeneous network, a platform (on the host 63) does not replicate until its health level becomes less than those of the hosts 55 and 62 at 2:00. When agents migrate from the host 63 to the hosts 55 and 62 at 12:00, platforms on the host 63 to the hosts 55 gain more energy from the agents and start replications.

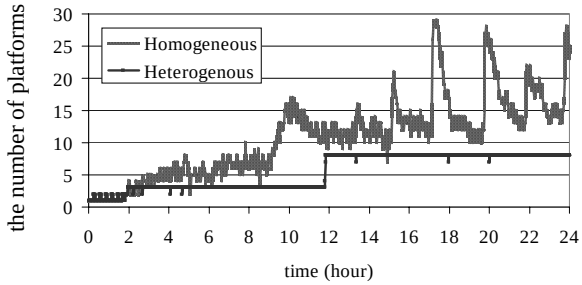

Fig. 4. The number of platforms

Fig. 5 shows the average distance between agents and users in network hop counts. In the homogeneous network, agents eventually work on the hosts where users reside on. In the heterogeneous network, the distance gradually (not rapidly) decreases because agent migration policy considers health level on neighboring hosts as well as users' locations.

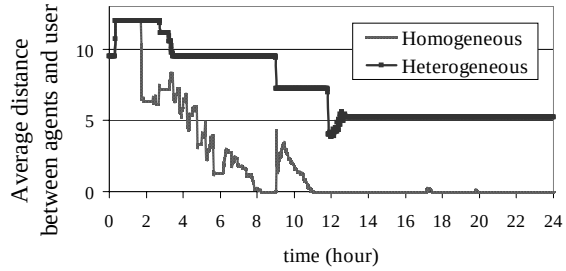

Fig. 5. Average distance between agents and users

Another observation in Fig. 5 is that platforms gradually move toward users, although platform replication policy does not consider users' locations. This is an example of symbiotic emergence. If replicated platforms are placed on hosts that agents want to migrate to (i.e. hosts closer to users), the platforms will survive. Otherwise, they will die because agents do not migrate to them and transfer energy to them. In a sense, agents indirectly instruct platforms where to replicate themselves. This results in a mutual benefit for agents and platforms. Agents can work closer to users and gain more energy from the users, and platforms gain more energy from agents.

Fig. 6 shows the quality of service (i.e. the average response time for agents to process service requests from users). In the first two hours, response time becomes very high in both homogeneous and heterogeneous networks, because agents have to store energy for a while to start replications. After that, agents start replications (see Fig 3); thereby decreasing response time dramatically. Also, as agents migrate towards users (see Fig. 4), response time decreases. Please note that response time include transmission latency between two network hosts (i.e. the closer agents work to users, the shorter their response time becomes). In the homogeneous network, response time drops to 15 seconds in two hours. In the heterogeneous network, response time drops to 5 seconds in four hours. Response time is shorter in the heterogeneous network because agents work on resource-rich hosts, where service requests are processed faster than on resource-poor hosts. After 14:00, response time becomes 0.5 second in the homogeneous network because agents work on the hosts where users reside on. In the heterogeneous network, agents decide to work on resource-rich hosts instead of the hosts where users reside on.

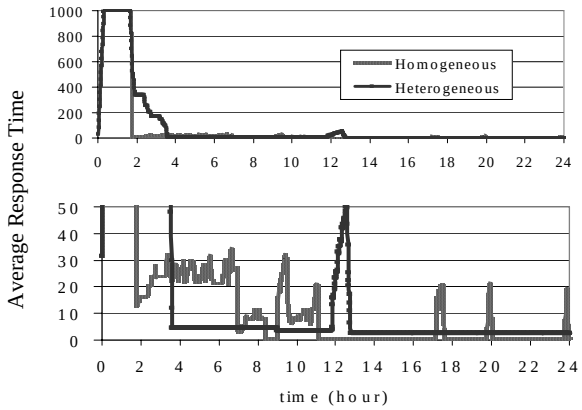

Fig.6 Response time for users

Fig. 7 shows resource efficiency. It is measured as (the total number of user requests processed by agents) / (the total amount of resources consumed by agents and platforms). The heterogeneous network generally maintains higher resource efficiency than the homogeneous one because the number of platforms in the heterogeneous network is much less than that in the homogeneous one (see Fig. 4). In the heterogeneous network, platforms tend to work on resource-rich hosts, and they can operate more agents than resource-poor hosts.

Fig. 8 shows how workload (i.e. service requests) is distributed over platforms. Load balancing index (LBI) is measured with the Equation 1 (LBI is a standard deviation of x).

$$LBI = \sqrt{\frac{\Sigma(x-\mu)^2}{N}} \qquad (1)$$

x indicates (the number of messages processed by agents) / (resource utilization on each platform). μ represents the expected average of x, which is measured as (the total number of messages processed by agents) / (the total amount of resource utilization on all platforms) / (the number of platforms; N). Fig. 8 shows agents and platforms gradually in

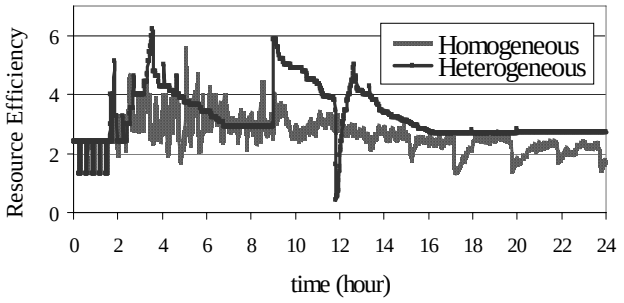

Fig. 7 Resource Efficiency

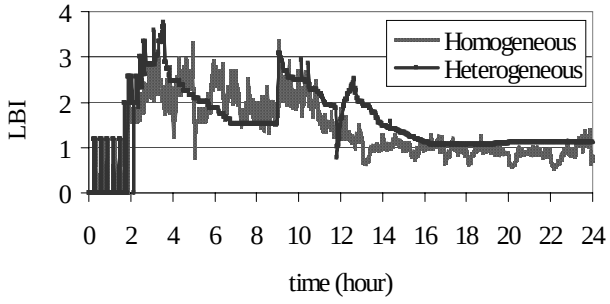

Fig. 8 Load Balancing Index

crease the degree of load balancing. This is an example of symbiotic emergence. Agent migration behavior policy encourages agents to move toward platforms running on healthier hosts. Platform replication behavior policy encourages platforms to replicate themselves on healthier hosts. As a result, service requests are processed by agents that are spread over the platforms running on healthy hosts. This contributes to balance workload per platform, although agent migration policy and platform replication policy do not consider agent population, platform population nor load balancing. This results in a mutual benefit for both agents and platforms. Platforms help agents decrease response time by making more resources available for them. Agents help platforms to keep their stability by distributing workload on them (i.e. by avoiding excessive resource utilization on them).

V. Related Work

This work is an extension to the Bio-Networking Architecture [4, 7]. In the Bio-Networking Architecture, agents are designed as biological entities, and they achieve service adaptation in a decentralized and collective manner. However, platforms are static and non-biological entities. Since they do not dynamically change their population and locations, they cannot achieve resource adaptation. In SymbioticSphere, both agents and platforms are biological entities, and they achieve both service adaptation and resource adaptation in a decentralized, collective and symbiotic manner.

Resource Broker [8] proposes a resource adaptation mechanism for grid systems. In this mechanism, a centralized system component monitors heterogeneous networks where different hosts have different levels of stability and different resource availability. Given monitored environment conditions, the mechanism adapts resource allocation for grid applications. Unlike Resource Broker, SymbioticSphere focuses on service adaptation as well as resource adaptation with decentralized agents and platforms.

[9] and [10] propose generic adaptation frameworks for grid systems. They can be used to achieve both service adaptation and resource adaptation. In these frameworks, centralized system components store the current environment conditions, and decide which adaptation strategy to execute against the monitored conditions. In contrast, SymbioticSphere does not assume any centralized system components. Each of agents and platforms collects and stores environment conditions, and autonomously decide which behavior to invoke.

The concept of energy in SymbioticSphere is similar to money in economy. MarketNet [11] applies the concept of money to address market-based access control for network applications. Instead of access control, SymbioticSphere currently focuses on scalability and adaptability of network systems (network applications and middleware platforms).

VI. Concluding Remarks

This paper overviews SymbioticSphere, and presents how it implements biological concepts and mechanisms. Simulation results show SymbioticSphere allows network systems to collectively scale and adapt to dynamic environment changes in an autonomous and decentralized manner.

References

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

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

[3] P. Chaprasert and J. Suzuki, "SymbioticSphere: A Biologically-inspired Network Architecture for Autonomic Grid Systems," In *Proc. of the 4th IASTED CIIT*, October 2005.

[4] J. Suzuki and T. Suda, "A Middleware Platform for a Biologically-inspired Network Architecture Supporting Autonomous and Adaptive Applications" In *IEEE J. on Selected Areas in Comm.* February 2005.

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

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

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

[8] A. Othman, P. Dew, K. Djemame, I, Gourlay, "Adaptive Grid Resource Brokering," In *Proc. of IEEE Int'l Conference on Cluster Computing*, Dec. 2003

[9] S. Cheng, D. Garlan, B. Schmerl, P. Steenkiste, and N. Hu, "Software Architecture-based Adaptation for Grid Computing," In *Proc. of IEEE Conference on High Performance Distributed Computing*, July 2002.

[10] K Shirose, S Matsuoka, H Nakada, and H Ogawa, "Autonomous Configuration of Grid Monitoring Systems," In *IEEE SAINT*, January 2004.

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