An Immunologically-inspired Autonomic Framework for Self-Organizing and Evolvable Network Applications

CHONHO LEE and JUNICHI SUZUKI
University of Massachusetts, Boston

Network applications are increasingly required to be autonomous, scalable, adaptive to dynamic changes in the network and survivable against partial system failures. Based on the observation that various biological systems have already satisfied these requirements, this paper proposes and evaluates a biologically-inspired framework that makes network applications to be autonomous, scalable, adaptive and survivable. With the proposed framework, called iNet, each network application is designed as a decentralized group of software agents, analogous to a bee colony (application) consisting of multiple bees (agents). Each agent provides a particular functionality of a network application, and implements biological behaviors such as reproduction, migration, energy exchange and death. iNet is designed after the mechanisms behind how the immune system detects antigens (e.g., viruses) and produces specific antibodies to eliminate them. It models a set of environment conditions (e.g., network traffic and resource availability) as an antigen and an agent behavior (e.g., migration) as an antibody. iNet allows each agent to autonomously sense its surrounding environment conditions (an antigen) to evaluate whether it adapts well to the sensed environment, and if it does not, adaptively perform a behavior (an antibody) suitable for the environment conditions. In iNet, a configuration of antibodies is encoded as a set of genes, and antibodies evolve via genetic operations such as crossover and mutation. Empirical measurement results show that iNet is lightweight enough. Simulation results show that agents adapt to dynamic and heterogeneous network environments by evolving their antibodies across generations. The results also show that iNet allows agents to scale to workload volume and network size and to survive partial link failures in the network.

Categories and Subject Descriptors: C.2.4 [**Computer-Communication Networks**]: Distributed System; H.4.0 [**Information Systems Applications**]: Communication Applications

General Terms: Design, Management
Additional Key Words and Phrases: Autonomic networking, Biologically-inspired networking, Artificial immune systems, Evolvable network applications

1. INTRODUCTION

As large-scale network applications such as data center applications and grid computing applications have been increasing in complexity and scale, they are increasingly required to address critical challenges such as *autonomy*–the ability to operate with minimal human intervention; *scalability*–the ability to scale to a large number of network hosts and users; *adaptability*–the ability to adapt to dynamic changes in network conditions (e.g., network

Author's address: Department of Computer Science, University of Massachusetts, Boston, 100 Morrissey Blvd. Boston, MA 02125. {chonho, jxs}@cs.umb.edu.
Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 20YY ACM 0000-0000/20YY/0000-0001 $5.00

traffic and resource availability); *survivability*–the ability to retain operation and performance despite partial system failures (e.g., network host/link failures) [Vasilakos et al. 2008; Dini et al. 2004; Sterritt and Bustard 2003; Rolia et al. 2000; Ranjan et al. 2002].

In order to address these requirements, the authors of the paper envision the network applications that exhibit self-organization[1] with inherent support of autonomy, scalability, adaptability and survivability. As inspiration for a new design paradigm to realize this vision, the authors observe that various biological systems have developed the mechanisms to meet the above requirements [Camazin et al. 2003]. For example, bees act autonomously, influenced by local conditions and local interactions with other bees. A bee colony can scale to a huge number of bees because all activities of the colony are carried out without centralized control. 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 flowers. When the hive is full of honey, bees rest in the hive. A bee colony can survive massive attacks by predators because it does not depend on any single bee, even on the queen bee. In fact, these desirable characteristics of a bee colony are not present in any single bee. Rather, they emerge from the collective actions and interactions of bees in the colony. Based on this observation, the authors of the paper believe that, if network applications are designed after certain biological principles and mechanisms, they may be able to meet the above requirements (i.e., autonomy, scalability, adaptability and survivability).

BEYOND[2] is an architecture that applies biological concepts and mechanisms to design network applications. Each network application is designed as a decentralized group of software agents. This is analogous to a bee colony (an application) consisting of multiple bees (agents). Each agent implements a functional service and follows biological behaviors such as reproduction, replication, migration, energy exchange and death.

This paper focuses on a key component in BEYOND: a biologically-inspired adaptation mechanism for network applications. The proposed mechanism, called iNet, is designed after the mechanisms behind how the immune system specifically produces antibodies to eliminate antigens (e.g., viruses) and how it evolves antibodies to react to a massive number of antigens. iNet models a set of environment conditions (e.g., network traffic and resource availability) as an antigen and an agent behavior (e.g., migration) as an antibody. Each agent contains its own immune system, and a configuration of the agent's antibodies defines its *behavior policy*, which determines when to invoke which behavior. iNet allows each agent to autonomously sense its surrounding environment conditions (an antigen) to evaluate whether it adapts well to the sensed conditions, and if it does not, adaptively perform a behavior (an antibody) suitable for the conditions. For example, agents may invoke the replication behavior on the network hosts that accept a large number of user requests for their services. This leads to the adaptation of agent population; agents can improve their throughput. Also, agents may invoke the migration behavior to move toward the network hosts that receive a large number of user requests for their services. This leads to the adaptation of agent locations; agents can improve their response time.

In iNet, a configuration of antibodies (i.e., behavior policy) is encoded as a set of *genes*. Agents evolve their antibody configurations so that the configurations become fine-tuned

[1]Self-organization is a process in which a system's internal components autonomously react to environmental changes, interact with each other and create an ordered/stable state (equilibrium point) without being guided by any outside sources [Camazin et al. 2003; Gershenson and Heylighen 2003].

[2]Biologically-Enhanced sYstem architecture beyond Ordinary Network Designs

to the current and even unexpected environment conditions. This evolution process occurs via genetic operations such as mutation and crossover, which alter antibody configurations (genes) during agent reproduction and replication. Evolution frees agent developers from anticipating all possible environmental changes and tuning their agents' antibody configurations (behavior policies) to the changes at design time. This can significantly simplify the implementation of agents.

This paper evaluates iNet through empirical and simulation studies. Empirical evaluation results show that iNet allows agents to efficiently sense their surrounding environment conditions and select a behavior suitable for the condition. Simulation results show that iNet allows agents to adapt to dynamic and heterogeneous network environments by evolving their antibody configurations (behavior policies) across generations. The results also show that iNet allows agents to scale to workload volume and network size and to survive partial link failures in the network.

This paper is organized as follows. Section 2 overviews the BEYOND architecture. Section 3 describes the design and implementation of iNet. Section 4 shows a series of empirical and simulation results to evaluate iNet. Sections 5 and 6 conclude with some discussion on related work and future work.

2. THE BEYOND ARCHITECTURE

This section overviews the design principles that BEYOND applies (Section 2.1), and presents the structure and behaviors of each agent (Section 2.2).

2.1 Design Principles

In BEYOND, agents are designed based on the five principles described below.

(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 BEYOND, there are no central entities to control and coordinate agents so that they can be scalable, survivable and simple by avoiding a single point of performance bottlenecks [Minar et al. 1999] and failures [Albert et al. 2001] and by avoiding any central coordination in deploying agents [Cabri et al. 2000].

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

(3) **Emergence:** In biological systems, collective (group) behaviors emerge from local interactions of autonomous entities [Camazin et al. 2003]. In BEYOND, agents only interact with nearby agents. They behave against dynamic changes of environment conditions such as user demands and resource availability. Through collective behaviors and interactions of individual agents, desirable system characteristics (e.g., load balancing and resource efficiency) emerge in a swarm of agents.

(4) **Lifecycle:** Biological entities strive to seek and consume food for living. Similarly, in BEYOND, agents 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., network bandwidth and memory space). The abundance or scarcity of stored energy in agents affects their lifecycle. For example, an abundance of stored energy indicates higher demand to an agent; thus, the agent may be de-

signed to favor reproduction or replication to increase its availability. A scarcity of stored energy (i.e., an indication of lack of demand) causes death of the agent.

(5) **Evolution:** In addition to individual adaptation, in which individual biological entities behave according to environmental changes, the entities evolve as a species to increase the fitness to the environment across generations. In BEYOND, as described above, individual agents adapt to environmental changes in the network by invoking their behaviors. In addition, agents collectively evolve their genes (behavior policies) by generating behavioral diversity and executing natural selection. Behavioral diversity means that different agents possess different behavior policies (genes). This is generated via genetic operations (e.g., mutation and crossover) during replication and reproduction. Natural selection is triggered with agents' energy levels. It retains the agents whose energy levels are high (i.e., the agents that have beneficial/effective behavior policies, such as moving toward a user to gain energy) and eliminates the agents whose energy levels are low (i.e., the agents that have detrimental/ineffective behavior policies, such as moving too often). Through successive generations, effective behavior policies become abundant in an agent species while ineffective ones become dormant or extinct. This allows agents to adapt to dynamic network environments.

2.2 Agent Structure and Behaviors

Each agent consists of *attributes*, *body* and *behaviors*. Attributes carry descriptive information regarding an agent, such as agent ID, energy level, the description of a service the agent provides, and the cost of a service (in energy unit) that the agent provides. Body implements the functional service that an agent provides. For example, an agent may implement a web service in a data center, while another agent may implement a scientific simulation model in a grid computing system. Behaviors implement the actions inherent to all agents. This paper focuses on the four behaviors described below.

(1) **Migration:** Agents may move from one network host to another. For example, some agents may move to the hosts near users to gain more energy from them. Others may move to the hosts whose resource availability is higher, in order to gain more resources.

(2) **Replication:** Agents may make a copy of themselves. A replicated (child) agent is placed on the host that its parent agent resides on, and it inherits the parent's antibody configuration (i.e., behavior policy) as well as the half amount of the parent's energy level. Mutation may occur on the inherited antibody configuration.

(3) **Reproduction:** Agents may produce their offspring with other agents (mating partners). A reproduced (child) agent is placed on the host that operates a parent that invokes the reproduction behavior. It inherits antibody configurations (i.e., behavior policies), via crossover, from its parent agents. Each parent also give a child agent the quarter amount of its energy level. Mutation may occur on the antibody configuration of a child agent.

(4) **Death:** Agents die due to energy starvation. If an agent cannot balance its energy expenditure with its energy gain, the agent cannot pay for the resources it needs; thus, it dies from lack of energy. When an agent dies, all resources allocated to the agent are released.

Each agent expends a certain amount of energy to invoke its behaviors except the death behavior. The behavior cost is constant for all agents.

3. THE INET ARTIFICIAL IMMUNE SYSTEM

This section overviews how the natural immune system works (Section 3.1) and describes how iNet is designed after the natural immune system (Section 3.2).

3.1 Natural Immune System

The natural immune system adaptively regulates the body against dynamic environmental changes such as antigen invasions. Through a number of interactions among various white blood cells (e.g., macrophages and lymphocytes such as T-cells and B-cells) and molecules (e.g., antibodies), the immune system evokes two responses to antigens: *T-cell activation* and *B-cell activation* responses.

In the T-cell activation response, the immune system performs self/non-self discrimination. This response is initiated by macrophages. Macrophages move around the body to ingest antigens and present them to T-cells. T-cells are produced in thymus though the negative selection. In the negative selection process, thymus removes the T-cells that strongly react to the body's own (self) cells. The remaining T-cells are used as detectors to identify foreign (non-self) cells. When a T-cell detects a non-self cell presented by a macrophage, the T-cell secretes chemical signals to induce the B-cell activation response.

In the B-cell activation response, B-cells are activated by T-cells. Some of the activated B-cells strongly react to an antigen, and they produce the antibodies that specifically kill the antigen. Antibodies form a network and communicate with each other [Jerne 1984]. This immune network is structured with stimulation and suppression relationships among antibodies. Through the interactions with these relationships, antibodies dynamically change their populations, thereby changing immune network structure. Thus, immune response is offered by multiple types of antibodies, although a single type of antibody (the best matched with an antigen) may play the dominant role. The immune network also regulates the quantitative balance of antibodies. For example, the population of specific antibodies rapidly increases following the detection of an antigen and, after eliminating the antigen, decreases again.

In order to react a massive number of antigens, the immune system needs to be able to generate various types of antibodies. A primary repertoire of antibodies is approximately 10^9 using immune genes. B-cells can increase this repertoire further by mutating and recombining immune gene segments so that antibodies can bind an unlimited number of antigens [Berek 2005].

3.2 Design and Implementation of iNet

The iNet artificial immune system consists of the environment evaluation (EE) facility and behavior selection (BS) facility, which implement the the T-cell activation response and B-cell activation response, respectively (Figure 1). The EE facility allows an agent to continuously sense a set of the current environment conditions as an antigen and classify the antigen to *self* or *non-self*. A self antigen indicates that the agent adapts to the current environment conditions well, and a non-self antigen indicates it does not. When the EE facility detects a non-self antigen, it activates the BS facility. The BS facility allows an agent to choose a behavior as an antibody that specifically matches with the detected non-self antigen.

3.2.1 *Environment Evaluation Facility.* The EE facility performs two steps: initialization and self/non-self classification. The initialization step produces detectors (T-cells)

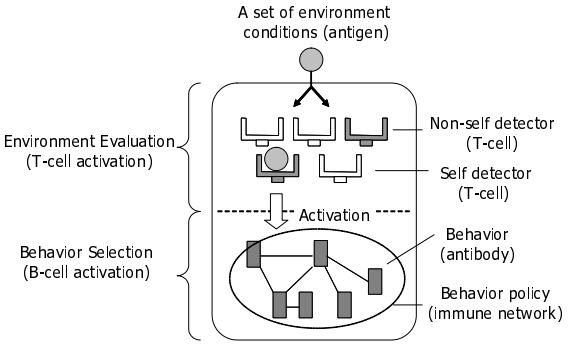

Fig. 1. iNet Architecture

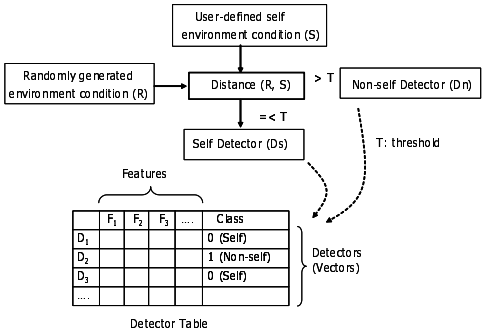

Fig. 2. Initialization Step in the EE facility

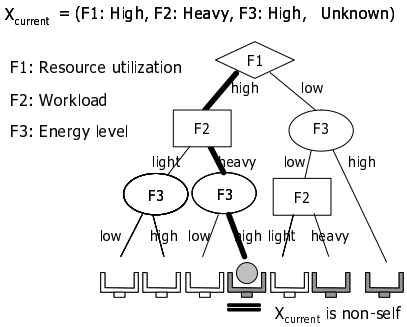

Fig. 3. An Example Decision Tree

that identify self and non-self antigens. Each antigen is represented as a feature vector $X = (F_1, F_2, ..., F_n, C)$, which consists of a set of environment conditions (or features: F_i) and a class value (C). Each environment condition, F_i, has a value. C indicates whether a given antigen (i.e., a set of environment conditions) is self (0) or non-self (1). For example, an antigen may be represented as $X_{current} = ((Low : Resource\ Utilization, Low : Workload), 0)$, if it senses resource utilization and workload (the number of user requests) on the local host.

The initialization step in the EE facility is designed after the negative selection process in the immune system (Figure 2). As the immune system randomly generates T-cells first, the EE facility generates detectors (feature vectors) randomly. Then, the EE facility separates the detectors into self detectors, which closely match with self antigens, and non-self detectors, which do not closely match with self antigens. This separation is performed via similarity measurement between randomly generated feature vectors (R) and self antigens (S) that human administrators supply. Similarity is measured based on the Euclidean distance between the two feature vectors (R and S). After the vector matching, both self and non-self detectors are stored in the feature table (Figure 2)[3].

The second step in the EE facility is self/non-self classification of an antigen (a set of current environment conditions). It is performed with a decision tree built from detectors in the feature table. Figure 3 shows an example decision tree that considers three features

[3]The natural immune system removes non-self detectors during the negative selection process. However, in iNet, both self and non-self detectors are used to perform self/non-self classification.

(environment conditions: F_1, F_2 and F_3). Each node in the tree represents a feature. Based on the feature values in a given antigen ($X_{current}$), the EE facility travels through tree branches from the root, and classifies the antigen to self or non-self according to the leaf node that the tree traversal reaches. In Figure 3, $X_{current}$ is classified to non-self as a result of a tree traversal through F_1, F_2 and F_3. If the EE facility classifies an antigen to non-self, it activates the BS facility so that an agent can adapt to the current environment conditions by selecting an appropriate behavior.

The reasons for using a decision tree as an antigen classifier are implementation simplicity and algorithmic efficiency. Decision trees perform classification much faster than other algorithms such as clustering, support vector machine and Markov model algorithms [Mitchell 1997]. The efficiency of classification is one of the most important requirements in iNet because each agent periodically performs self/non-self classification at runtime. A decision tree is built with the information gain technique [Mitchell 1997] that determines which features to be located at each layer in a tree while minimizing the tree's height.

3.2.2 *Behavior Selection Facility.* The BS facility selects an antibody (i.e., agent's behavior) suitable for a non-self antigen (i.e., environment conditions). Each antibody is structured as shown in Figure 4. It consists of *paratope*, a precondition (an environment condition) under which it is selected; Behavior ID; and *idiotope*, relationships to other antibodies. Antibodies are linked with each other using stimulation and suppression relationships (see Section 3.1). Each antibody has its own concentration value, which represents its population. The BS facility identifies candidate antibodies (behaviors) suitable for a given non-self antigen (environment conditions), prioritizes them based on their concentration values, and selects the most suitable one from the candidates. When prioritizing antibodies (behaviors), stimulation relationships between them contribute to increase their concentration values, and suppression relationships contribute to decrease it. Each relationship has an affinity value, which indicates the degree of stimulation or suppression.

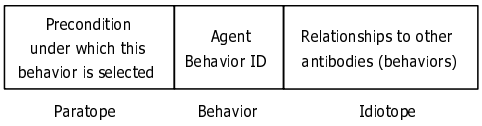

Fig. 4. An Antibody Structure

Figure 5 shows a generalized immune network of antibodies. The antibody i stimulates M antibodies and suppresses N antibodies. m_{ji} and m_{ik} denote affinity values between antibody j and i, and between antibody i and k, respectively. m_i is an affinity value between an antigen and antibody i. The concentration of antibody i, denoted by a_i, is calculated with the following equations.

$$\frac{dA_i(t)}{dt} = \left(\frac{1}{N} \sum_{j=1}^{N} m_{ji} \cdot a_j(t) - \frac{1}{M} \sum_{k=1}^{M} m_{ik} \cdot a_k(t) + m_i - k \right) \cdot a_i(t) \tag{1}$$

$$a_i(t) = \frac{1}{1 + exp(0.5 - A_i(t))} \tag{2}$$

In Equation (1), the first and second terms in a bracket denote the stimulation and suppression from other antibodies. m_{ji} and m_{ik} are positive in between 0 and 1. m_i is 1 when antibody i is stimulated directly by an antigen, otherwise 0. k denotes the dissipation factor

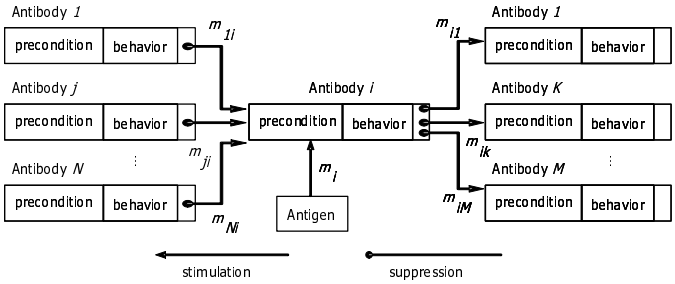

Fig. 5. A Generalized Immune Network

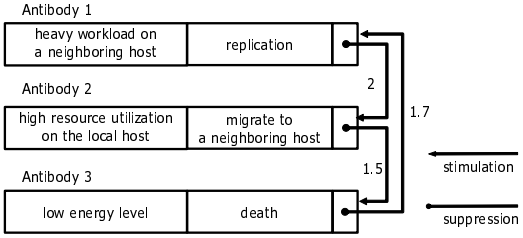

Fig. 6. An Example Immune Network

representing the natural death of an antibody. Equation (2) is a sigmoid function used to squash the $A_i(t)$ value between 0 and 1.

Every antibody's concentration is calculated 200 times repeatedly. This repeat count is obtained from a previous simulation experience [Lee and Suzuki 2006]. If no antibody exceeds a predefined threshold during the 200 calculation steps, the antibody whose concentration value is highest is selected (i.e., winner-takes-all selection). If one or more antibodies' concentration values exceed the threshold, an antibody is selected based on the probability proportional to all agents' concentration values (i.e., roulette-wheel selection).

Figure 6 shows an example immune network. The network contains three antibodies, which represent the replication, migration and death behaviors. It also contains three relationships among the three antibodies. Antibody 1 represents the replication behavior and specifies that the behavior is invoked when workload is heavy on a neighboring host. Antibody 1 stimulates antibody 2 and suppresses antibody 3. Now, suppose a (non-self) antigen indicates that (1) workload is heavy on a neighboring host and (2) energy level is low. This antigen stimulates antibodies 1 and 3 simultaneously. Their concentration values increase, and antibody 2's concentration value becomes the highest because the stimulation affinity from antibody 1 to 2 is greater than the suppression affinity from antibody 2 to 3. As a result, the BS facility would select antibody 2; thus, an agent would invoke the migration behavior to move to a neighboring host.

3.2.3 *Evolution of Antibody Configurations.* As described in Section 3.1, the natural immune system diversifies antibodies by mutating immune genes so that antibodies can react to unanticipated antigens. Similarly, iNet diversifies behavior policies via gene operations (mutation and crossover) so that agents can adapt to unanticipated environment conditions. In iNet, each agent encodes and possesses its behavior policy as a set of genes. The genes are represented as a sequence of numbers (genotype), as shown in Figure 7. When a new agent is born through a replication or reproduction process, it interprets an

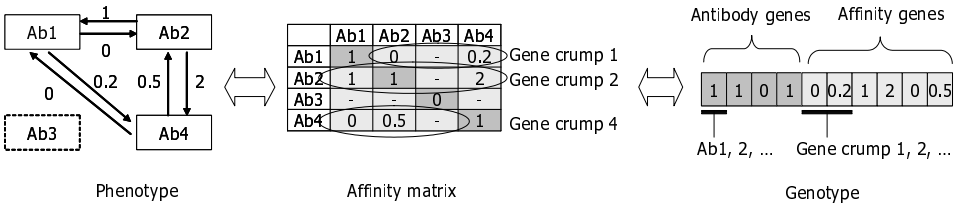

Fig. 7. Agent Genes (Genotype and Phenotype)

inherited genotype and form an immune network (behavior policy) as a phenotype. An *affinity matrix* is used to map a genotype to phenotype, and vice versa. The agent genotype consists of the *antibody genes*, which specify the presence of antibodies, and the *affinity genes*, which specify relationships among antibodies and their affinity values.

When an agent invokes the reproduction behavior, it searches mating partner candidates from its local and neighboring hosts. This search process is performed based on the topological distance (hop count) from the agent. The agent uses Equation 3 to obtain its mating partner candidates by randomly selecting $M\%$ of the agents running on k-hops-away neighboring hosts. (γ is a constant.) All (100%) of the agents running on the local (i.e., 0-hop-away) host are available as mating partner candidates, and the number of available candidates decreases as k increases. If no candidates are found, an agent performs the replication behavior instead of the reproduction behavior.

$$M = \frac{100}{k^{\gamma}} \tag{3}$$

Among available candidates, a mating partner is selected by ranking them. iNet uses a domination ranking mechanism [Deb 2001]. Agents are ranked with three objectives: (1) energy utility, (2) behavior selection efficiency and (3) the number of alive children (offspring agents). In all of these three objectives, the higher, the better. Energy utility is the rate of an agent's total (lifetime) energy gain to its total (lifetime) energy expenditure (Equation 4). Behavior selection efficiency indicates an agent's energy gain per behavior invocation; that is, how an agent effectively invokes a behavior to gain energy. It is updated as an exponentially-weighted moving average (EWMA) each time an agent invokes a behavior (Equation 6). $EnergyLevel(t)$ is the current energy level, and $EnergyLevel(t-1)$ is the one at the time of the previous behavior invocation. EWMA is used to smooth out short-term minor oscillations in the data series of $\Delta EnergyLevel$. It places more emphasis on the long-term transition trend of $\Delta EnergyLevel$; only significant changes in $\Delta EnergyLevel$ have the effects to change O_2. The α value is a constant to control the responsiveness of EWMA against the changes of $\Delta EnergyLevel$. The third objective is the number of alive children (Equation 6). It indicates how many alive offspring agents still exist (or survive) out of the ones an agent has ever created through replication and reproduction.

$$O_1(t) = \frac{Total\ energy\ gain}{Total\ energy\ expenditure} \tag{4}$$

$$O_2(t) = (1-\alpha) * O_2(t-1) + \alpha * \Delta EnergyLevel \tag{5}$$
$$where\ \Delta EnergyGain = EnergyLevel(t) - EnergyLevel(t-1)$$

$$O_3(t) = The\ number\ of\ alive\ children\ (t) \tag{6}$$

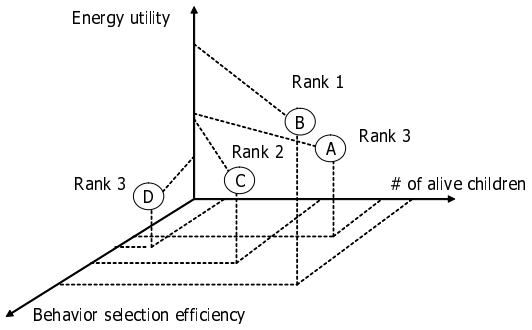

Fig. 8. An Example of Domination Ranking in a Objective Space

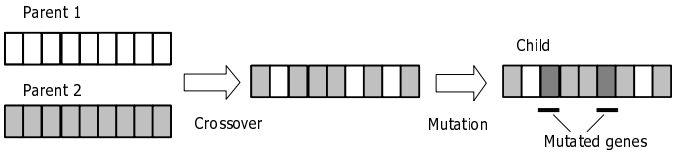

Fig. 9. An example Genetic Operations

All mating partner candidates are plotted on a three dimensional space whose axises represent the objectives described above. Then, each candidate is evaluated whether it is *dominated* by another candidate. An candidate is considered to be dominated if another candidate outperforms it in all of three objectives. Figure 8 shows an example of ranking four different agents (Agent A to D). Agent B dominates all the other agents in all three objectives; it is called *non-dominated* and given Rank 1. Agent C is dominated by Agent B; however, it dominates Agent D; it is given Rank 2. Agents A and D dominate no other agents; they are given Rank 3. As a result, Agent B (non-dominated agent) is selected as a mating partner. If multiple non-dominated agents are available, one of them is randomly selected as a mating partner.

In reproduction, two parents contribute their genes (behavior policies), via crossover, to a child agent. The amount of their gene contributions follow the ratio of the number of outperforming objectives. For example, in Figure 9, when Agent 1 outperforms Agent 2 in two objectives and, the ratio of the number of outperforming objectives is 2:1 between Agent 1 and 2. Thus, Agent 1 contributes 2/3 of its genes to a child agent, and Agent 2 contributes the rest (1/3). In replication, a parent agent contributes its whole genes to a child agent. Both in reproduction and replication, mutation may occur on a child agent's genes in a certain probability (mutation rate).

4. EVALUATION

This section evaluates iNet through empirical experiments (Section 4.1) and simulations (Sections 4.2 to 4.5).

4.1 Empirical Measurement of the EE and BS Facility

This section empirically evaluates the overhead of the EE and BS facilities. All measurements were conducted with a Windows XP PC that has a 2.0 GHz Intel Celeron CPU and 512MB memory space. Table I shows the overhead of self/non-self classification in the EE facility ($T_{classify}$). It indicates how long it takes for the EE facility to classify an antigen

(a set of environment conditions) into self or non-self. Each agent incurs this classification overhead at runtime. However, the overhead is small enough and acceptable for most network applications. Note that seven features are used in each of all subsequent experiments.

Figure 10 shows the overhead of the BS facility (T_{BS}). It indicates how long it takes for the BS facility to select an antibody (behavior) suitable for a given antigen. The overhead of the BS facility is larger than that of the EE facility, and it increases exponentially as the number of antibodies grows. However, the BS facility does not work periodically; it is activated only when an agent does not adapt well to the current environment conditions.

The Number of Features	3	4	5	6	7	...	10
$T_{classify}$ $(msec)$	1.5	3.0	3.0	3.0	3.0	...	4.5

Table I. Overhead of Self/Non-self Classification in the EE Facility

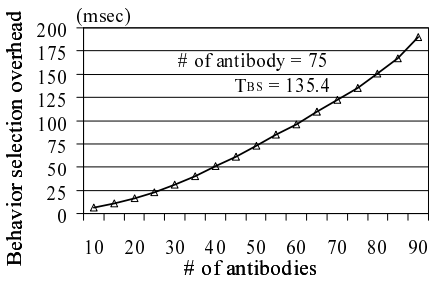

Fig. 10. Behavior Selection Overhead

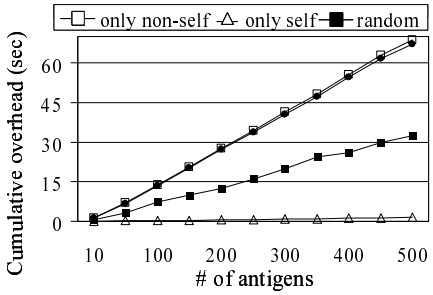

Fig. 11. Cumulative Overhead

For evaluating the impact of the EE facility on the efficiency of iNet, Figure 11 shows the cumulative overhead of iNet that repeatedly executes both the EE and BS facilities against multiple antigens. This overhead is measured with three scenarios described below.

(1) **Self environment conditions only:** The EE facility receives self environment conditions only. This scenario emulates a static network where its environment conditions do not dynamically change and agents always adapt well to them.

(2) **Non-self environment conditions only:** The EE facility receives non-self environment conditions only. This scenario emulates a dynamic network where its environment conditions change and agents always need to adapt to them with the BS facility.

(3) **Random environment conditions:** The EE facility randomly receives self and non-self environment conditions. It receives self environment conditions at the probability of 50%. This scenario emulates a dynamic network where its environment conditions dynamically change and agent need to adapt to them at the probability of 50%.

In Scenario 1, iNet executes only the EE facility for every antigen because all given antigens are self. It takes 1.5 seconds to execute the EE facility for 500 antigens. In Scenario 2, iNet always executes both the EE and BS facilities for every antigen because all given antigens are non-self. It takes approximately 70 seconds to execute both facilities for 500 antigens. The overhead of the BS facility dominates this total overhead. In Scenario 3, iNet executes the EE facility for self antigens and both the EE and BS facilities for non-self antigens. Consequently, its overhead result is in between those in Scenarios 1 and 2. Figure 11 shows that the EE facility effectively avoids executing the BS facility when

it identifies self antigens, thereby eliminating the unnecessary overhead (and associated resource consumption) of the BS facility.

4.2 Simulation Configurations

This section shows a series of configurations used in simulations. All simulations were carried out on the BEYOND simulator[4]. (See also Appendix A.) In each simulation, iNet considers seven environment conditions listed in Table II. Also, iNet uses the parameter values shown in Table III. Figure 12 shows a simulated network consisting of 100 hosts connected in a 10x10 grid topology. Each agent implements a web service that receives a service request and returns an HTML file. Service requests travel from users to agents via user access point (Figure 12). This simulation study assumes that a single (virtual) user runs on the access point and sends service requests to agents.

Figure 13 shows how the user changes service request rate over time. This follows a workload trace of the www.ibm.com site [Chase et al. 2001]. The workload peaks 5,500 and 9,000 requests/minute in the morning and the afternoon, respectively. At the beginning of each simulation, four agents are deployed on randomly-selected hosts, and each agent's behavior policy is randomly configured.

4.2.1 *Performance Index.* To quantify agent performance, this simulation study uses the following performance index, which is a weighted sum of performance factors (p_i):

$$Performance\ Index = \sum w_i \cdot p_i \tag{7}$$

This simulation study considers the following four factors. Each factor value is non-negative between 0 and 1.

(1) **Response time (p_1):** The response time of an agent for a service request from the

[4]The current code base of the BEYOND simulator contains 16,492 lines of Java code.

Environment Condition (paratope)		
Condition	Description	Value
EnergyLevel	Energy level of an agent	High, Low
# of agents@localHost	The number of agents on the local host	Large, Small
# of agents@neighboringHost	The number of agents on a neighboring host	Large, Small
Workload@localHost	Workload at the local platform	Heavy, Light
Workload@neighboringHost	Workload at a neighboring platform	Heavy, Light
ResourceAvail@localHost	Resource availability on the local host	High, Low
ResourceAvail@neighboringHost	Resource availability on a neighboring platform	High, Low

Table II. Environment Conditions considered in Simulations

Parameter	Symbol	Value
Dissipation factor (Equation 1)	k	0.1
Distance factor (Equation 3)	γ	1.5
EWMA coefficient (Equation 6)	α	0.5
Weight value for each performance factor (Equation 7)	w_i	0.25
The minimum time for an agent to process a single request (Equation 8)	R	0.25
The maximum number of messages an agent can process per minute (Equation 10)	M_{max}	240

Table III. Parameter Values used in Simulations

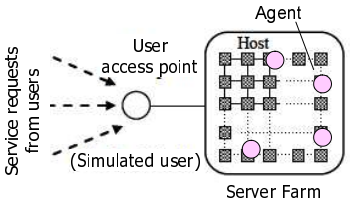

Fig. 12. A Simulated Network

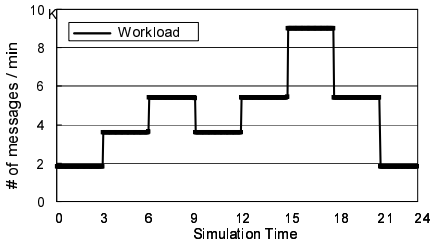

Fig. 13. Workload

user. R is the minimum time for each agent to process a single service request.

$$p_1 = \frac{R}{Response\ time} \tag{8}$$

(2) **Throughput (p_2):** This factor indicates how many service requests agents process.

$$p_2 = \frac{\#\ of\ service\ requests\ processed\ by\ all\ agents}{Total\ \#\ of\ service\ requests\ issued\ by\ the\ user} \tag{9}$$

(3) **Load distribution (p_3):** This factor indicates how workload is distributed over agents. m denotes the number of service requests that an agent processes in a unit time. μ_m denotes the expected number of service requests that each agent processes. M_{max} denotes the maximum number of service requests that an agent can process in a unit time.

$$p_3 = 1 - \frac{abs(m-\mu_m)}{M_{max}}\ where\ \mu_m = \frac{Total\ \#\ of\ service\ requests\ issued\ by\ the\ user}{Total\ \#\ of\ agents} \tag{10}$$

(4) **Resource utilization balance (p_4):** This factor indicates how resource utilization is distributed over hosts. r denotes the resource utilization rate on the local host that an agent resides on (0 to 1; 0% to 100%). This is measured as the ratio of the amount of resources consumed by agents on the host to the amount of resources available on the host. μ_r denotes the expected resource utilization rate on each of the hosts where agents run.

$$p_4 = 1 - abs(r-\mu_r)\ where\ \mu_r = \frac{The\ sum\ of\ resource\ utilization\ rate\ on\ all\ hosts}{\#\ of\ hosts\ that\ agents\ reside\ on} \tag{11}$$

4.2.2 *Mutation Rate and Range.* Two parameters, mutation rate (MR) and mutation range (MG), impact the evolution and adaptation of agents. MR indicates how often genes (behavior policies) are altered during replication and reproduction. RG is the value range to alter genes. In order to evaluate the impacts of different MRs on agent evolution, Figures 14 and 15 show how the average and maximum performance index change over time, respectively. With a lower MR, agents improve performance index more slowly; however, the improvement is more stable. With a higher MR, agents improve performance index more quickly; however, larger fluctuations exist in the improvement.

In order to evaluate how different RGs impact agent performance, Figures 16 and 17 show the transition of average and maximum performance index, respectively. With a smaller RG, agents improve performance index more slowly; however, the improvement is

Mutation Rate (MR)		Mutation Range (RG)	
0	0.01498	1	0.01576
0.3	0.01872	2	0.01872
0.6	0.01913	3	0.02094
1	0.02815	4	0.02698

Table IV. Sum of Squared Errors to a Linear Regression Line

more stable. With a higher RG, agents improve performance index more quickly; however, larger fluctuations exist in the improvement.

Figure 18 and Table IV focus on the fluctuations in the improvement of performance index. Figure 18 shows the sum of difference between two consecutive performance index ($\sum_t abs(performance\ index(t) - performance\ index(t-1))$).

For each line in Figures 14 and 16, linear regression was performed. Table IV depicts the difference between a regression line and an actual line as the sum of squared errors between them. Figure 18 and Table IV confirm that fluctuation becomes larger in the improvement of performance index as MR and RG grow.

For agents (network applications), both speed and stability are important in the improvement of performance index. By balancing this tradeoff, subsequent simulations use MR of 0.3 and RG of 2.

4.3 Agent Adaptability

This section shows a series of simulation results to evaluate agent adaptability. Section 4.3.1 evaluates how an iNet evolution process impacts the adaptability of agents. Section 4.3.2 shows how the EE facility contributes to improve the adaptability of agents.

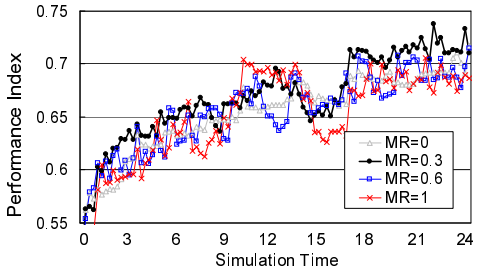

Fig. 14. Average Performance Index with Different Mutation Rates (RG = 2)

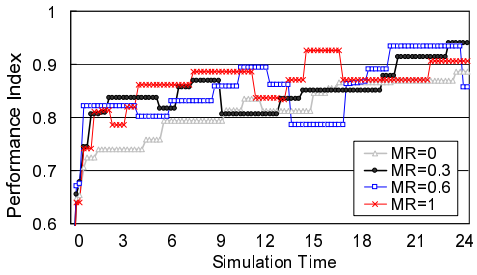

Fig. 15. Maximum Performance Index with Different Mutation Rates (RG = 2)

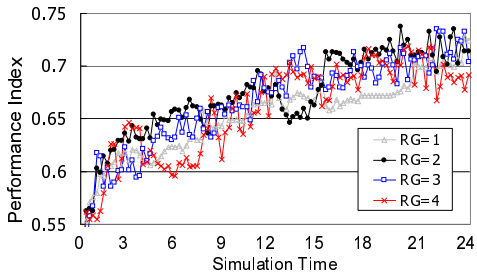

Fig. 16. Average Performance Index with Different Mutation Ranges (MR = 0.3)

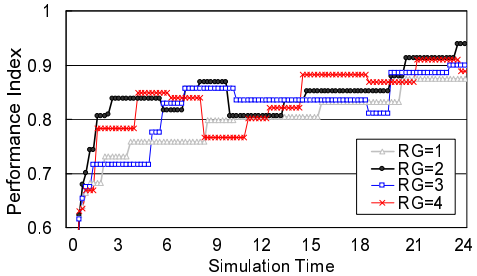

Fig. 17. Maximum Performance Index with different Mutation Ranges (MR = 0.3)

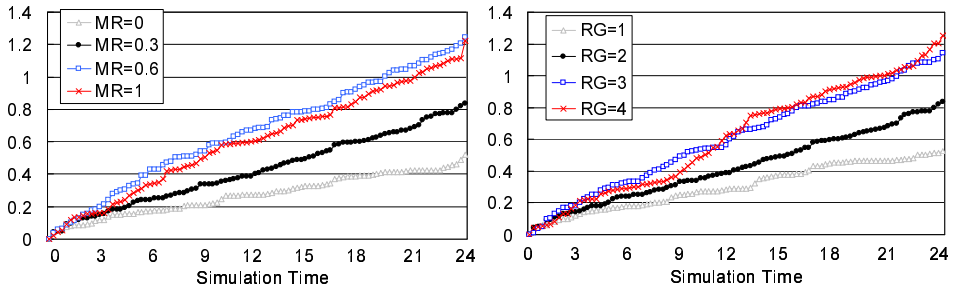

Fig. 18. Sum of Difference between Consecutive Performance Index

4.3.1 *Evaluation of iNet Evolution Process.* This section presents how iNet evolution process impacts the adaptability of agents by comparing simulation results with and without evolution in iNet. Figure 19 shows how agents adapt their population to workload changes depicted in Figure 13. With evolution, iNet allows agents to evolve and adapt their genes (behavior policies). Thus, as they gain energy by processing service requests, they properly perform the replication or reproduction behaviors to increase their population. They also properly perform the death behavior to decrease their population when workload decreases. On the other hand, without evolution, agents do not change their randomly-configured genes throughout a simulation. As a result, they fail to adapt their population to workload changes.

Figure 20 shows how agents adapt their throughput to workload changes. Evolvable agents autonomously maintain high throughput by dynamically adjusting their locations and population with the migration and reproduction behaviors. Without evolution, agents fail to adapt their throughput because they do not evolve their genes.

Figure 21 shows how agents reduce their response time for the user. At the beginning of a simulation, response time stays high because only four agents process 2,000 service requests a minute and the distance between the agents and the user is long. However, as evolvable agents accumulate enough energy from the user and perform the migration, replication and reproduction behaviors, they rapidly decrease their response time. When workload spikes at 3:00, response time increases up to 15 seconds; however, agents decrease it to approximately 3 seconds by adapting their locations and population. Since then, they maintain low response time even when workload spikes again at 6:00, and 12:00 and 15:00. Without evolution, agents cannot maintain low response time.

Figure 22 shows the distance (average number of hops) between agents and the user. At the beginning of a simulation, four agents are randomly deployed on the network, the distance tends to be large. However, evolvable agents gradually decrease it by moving toward the user. As discussed above, this adaptation of agent locations contributes to decrease response time for the user. Without evolution, agents do not adapt their locations throughout a simulation.

Figure 23 shows how workload is distributed over agents with the Load Balancing Index (LBI).

$$\textit{Load Balancing Index} = \sqrt{\frac{\sum(X_i - \mu)^2}{N}} \tag{12}$$

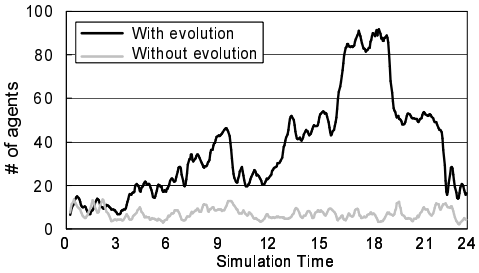

Fig. 19. Agent Population

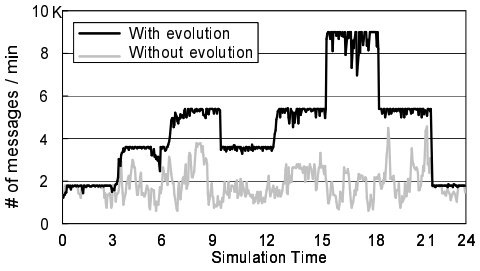

Fig. 20. Agent Throughput

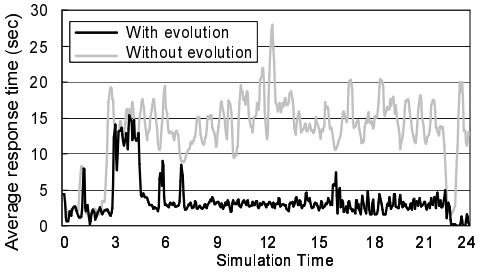

Fig. 21. Average Response Time

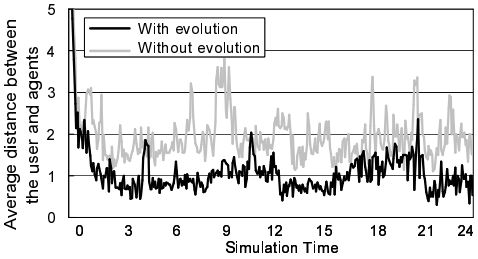

Fig. 22. Average Distance between Agents and the User

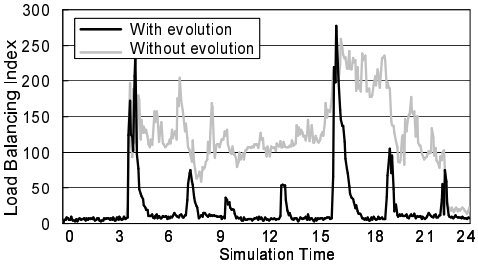

Fig. 23. Load Balancing Index

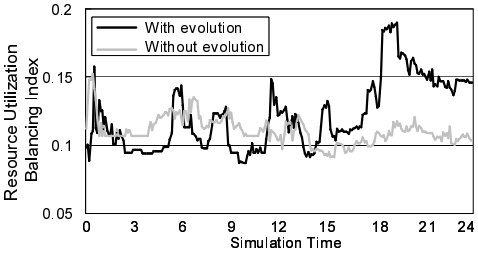

Fig. 24. Resource Utilization Balancing Index

N denotes the total number of agents. X_i denotes the number of service requests processed by agent i. μ denotes the expected X (i.e., the total number of service requests in the network divided by N). LBI represents the variance of the number of service requests allocated to agents. A lower LBI indicates higher workload distribution over agents.

When workload spikes at 3:00, 6:00 12:00 and 15:00, LBI increases because all requests are not distributed among agents although they attempt to replicate themselves to process them evenly. When workload drops at 9:00, 18:00 and 21:00, LBI increases because some agents become idle and process no requests. Despite these spikes, evolvable agents immediately decrease LBI by adjusting their population. Without evolution, agents fail to yield low LBI throughout a simulation.

Figure 24 indicates how resource utilization is distributed over hosts with Resource Utilization Balancing Index (RUBI).

$$\textit{Resource Utilization Balancing Index} = \sqrt{\frac{\sum(R_k - \mu)^2}{N}} \tag{13}$$

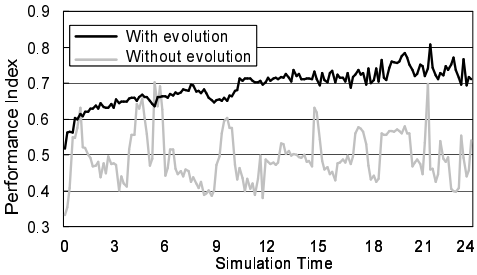

Fig. 25. Average Performance Index

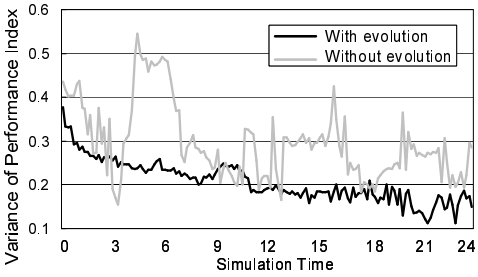

Fig. 26. Variance of Performance Index

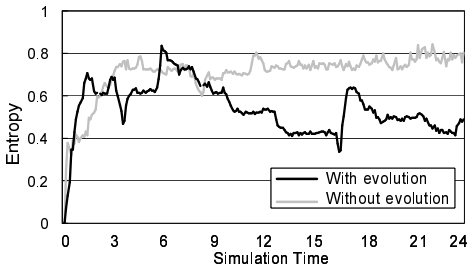

Fig. 27. Entropy: The Degree of Self-Organization

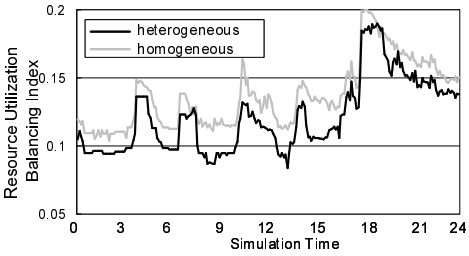

Fig. 28. RUBI in Homogeneous and Heterogeneous Networks

N denotes the number of active hosts, which are the hosts where agents are running. R_k denotes the resource utilization on host k. μ represents the expected R (the amount of resources consumed by all agents over the total amount of resources that the active hosts provide). RUBI represents the variance of resource utilization among active hosts. A lower RUBI indicates higher distribution of resource utilization.

When agents increase their population according to an an increase in workload, resource utilization grows on the hosts on which they reside. However, some of them migrate to neighboring hosts that offer higher resource availability. This means that agents strive to spread over hosts evenly; therefore, they decrease RUBI immediately after its spikes. Without evolution, agents fail to distribute resource utilization over hosts.

Figure 25 shows the average performance index of agents. (See also Equation 7.) While agents do not improve their performance with evolution disabled, they gradually improve it toward 0.8 with evolution enabled. Together with Figures 19 to 24, Figure 25 demonstrates that iNet allows agents to successfully evolve and adapt to dynamic network conditions.

Figure 26 shows the variance of performance index among agents. A lower variance indicates that agents yield performance index results more similarly. While agents do not decrease their performance index variance with evolution disabled, they converge it to a very low variance with evolution enabled. Together with Figure 25, Figure 26 demonstrates that iNet allows all agents to successfully evolve so that they yield similar and high performance results.

Figure 27 depicts the degree of self-organization of agents with the entropy metric. Entropy is measured based on the distribution of agents in the objective space. (See also Figure 8). It quantifies the disorderliness of agents in the objective space. A lower entropy indicates that agents are more ordered. It is said that agents are self-organizing when

they autonomously decrease their entropy [Gershenson and Heylighen 2003]. Entropy is calculated as follows:

$$Entropy = -\sum_{i} p_i \times log(p_i)$$

The entire objective space is divided to 27 cubes, and p_i denotes the probability that agents exist the i-th cube. The probability is measured as the number of agents in the i-th cube over the total number of agents.

As Figure 27 shows, entropy increases at the beginning of a simulation because replicated and reproduced agents yield different objective values. However, through evolution process, agents gradually decrease their entropy. This means that they autonomously yield similar objective values over time. Together with Figure 25, Figure 27 demonstrates that iNet allows all agents to successfully self-organize in the objective space through evolution so that they yield similar objective values and produce high performance results.

iNet allows agents to adapt to different environment on hosts, homogeneity and heterogeneity in terms of resource availability (e.g.,memory space). In order to evaluate how iNet agents effectively work on heterogeneous environment, the simulated network is assumed as a server farm consisting of 50 hosts with 64MB memory space and 50 hosts with 128MB memory space (of 10x10 grid topology). Figure 28 shows the trace of resource utilization balancing index (RUBI) on both homogeneous and heterogeneous environment. Agents successfully reduce RUBI to keep balancing the provided resource utilization. In the heterogeneous environment, RUBI becomes smaller than that of homogeneous environment becuase more agents start to invoke high resource migration (i.e., migration to a neighboring host which has higher resource availability) behavior. More agents run on hosts with more resources.

4.3.2 *Evaluation of the EE Facility.* In iNet, evolution process occurs on immune network configurations in the BS facility. The EE facility also contributes to save resource consumption and execution overhead for behavior selections; in addition, it contributes to improve the adaptability of agents. Without the EE facility, agents periodically monitors environment conditions and invokes one of behaviors regardless of whether the agent adapts well or not to the surrounding environment. This results in wasting resources (e.g., memory space and CPU cycle) caused by unnecessary behavior selection process. With the EE facility, agents first evaluate if they adapt to the current environment or not, then they execute the BS facility only when they do not adapt to the environment.

In order to evaluate the impact of the EE facility, two different types of agents-agents with and without the EE facility, i.e. *EE and BS* and *Only BS*-are compared. Similar to the previous set of simulation results (Section 4.3.1), the change of agent population, response time for the user, agent throughput, load balancing index (LBI) and resource utilization balancing index (RUBI) are measured according to the workload trace (See Figure 13) and shown in Figure 29, 30, 31, 33, and 34 respectively.

Both type of agents will improve their performance well compared to the agents without evolution (described in the previous subsection). However, it is shown that agents without the EE facility unnecessarily invoke some behaviors when they do not have to do. For example, around 7:00 in Figure 29, about 50 agents with the EE facility processed all service requests (about 5,500 messages per min) in timely manner. Yet, agents without the EE facility keep changing their population to about 70 by invoking replication or death

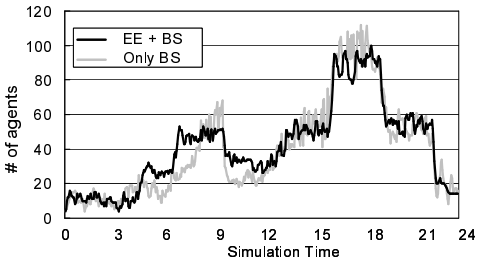

Fig. 29. Population of Agents

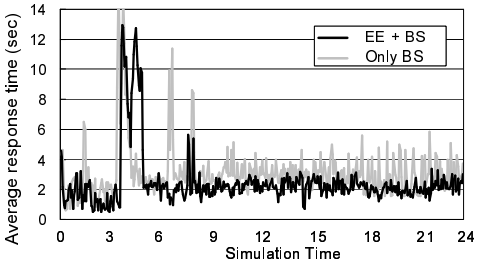

Fig. 30. Average Response Time for the User

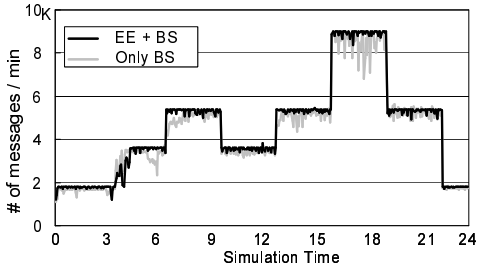

Fig. 31. Agent Throughput

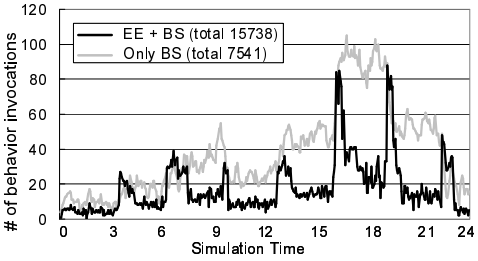

Fig. 32. The Number of Behavior Invocations

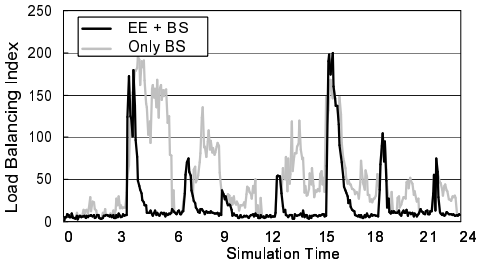

Fig. 33. Load Balancing Index

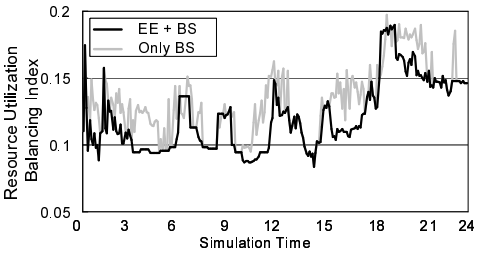

Fig. 34. Resource Utilization Balancing Index

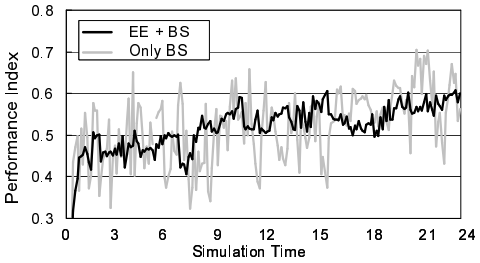

Fig. 35. Average Performance Index of Agents

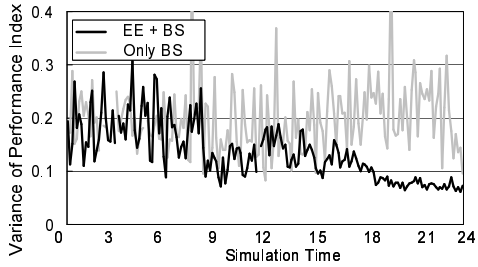

Fig. 36. Variance of Performance Index of Agents

behavior. Then, at 9:00, some of them die as soon as the workload drops because they are over-replicated and could not gain enough energy from the user. Besides, since agents without the EE facility cannot adapt their population in timely manner, they also do not adaptively reduce the response time for the user (shown in Figure 30). They might migrate to nodes far from the user or die unexpectedly.

Similarly, there are some timelag to improve the performance between two different

types of agents (i.e. with and without the EE facility). Agents with the EE facility achieves faster improvement of performance than agents without it; and, the fluctuation in the performance improvement becomes larger without the EE facility. In other words, plots (gray lines) for agents without the EE facility in Figures 30, 31, 33, and 34 are somehow unstable and swinging.

Because of the unnecessary behavior invocations, agents without the EE facility waste resources and spend additional execution overhead. Figure 32 shows the total number of behavior invocations of running agents during each simulation cycle, i.e., how many times agents invoke their behaviors. For agents without the EE facility, the number of behavior invocations is the exactly same as the number of agents because they monitor environment conditions and invoke one of behaviors each simulation cycle. Agents with the EE facility are likely to perform behaviors when the workload changes. In total, agents with the EE facility executes the BS facility 7541 times while agents without the EE facility executes the BS facility 15738 times during a simulation. Agents with the EE facility will save almost 47.9% of behavior invocation frequency.

Even though agents with the EE facility performs less behavior selections, they effectively improve their performance index. Figure 35 shows that the average performance index of agents without the EE facility is unstable, i.e. swinging, and Figure 36 shows that the variance for agents without the EE facility has not converged well. It follows that the optimal iNet configuration (genes) is successfully spread out to other surviving agents; thus, the EE facility also contribute for agents to adapt to the environment conditions well through generations.

4.4 Agent Scalability

This section shows how agents scale in different size of network in terms of the number hosts in a server cluster and the workload volume (i.e., the number of service requests per min). To evaluate the agents scalability, following two cases are compared as described in Figure 37. (i) 9K requests per min at peak in 100 (10x10) hosts, and (ii) 5K requests per min at peak in 9 (3x3) hosts. The workload trace follows one thirds of daily request rate at the www.ibm.com site in February, 2001 [Chase et al. 2001].

Figure 38 to 42 show the comparison results how agents adapt their population, improve their throughput, reduce the response time for the user, LBI, RUBI respectively to different scale of workload. Figure 43 and 44 show that agents gradually improve their performance index and the variance of performance index among alive agents by spreading out the configuration of immune network (i.e. genes).

Similar to the results in Section 4.3.1, agents successfully adapt their population and location according to workload changes in order to improve their throughput and response time for the user, etc. Because agents do not involve in any interventions from/to other controlling agents, agents are running in decentralized manner regardless of the size of network and workload volume.

4.5 Agent Survivability

This section presents the agents' survivability against link failures in server clusters. The Internet data center should not degrade its performance (e.g., response time for the user and throughput) and be able to recover from the any types of failures (e.g., unexpected link failures due to disasters) immediately. The simulation results show that agents with iNet adapt their population and location against link failures and contribute to improve response

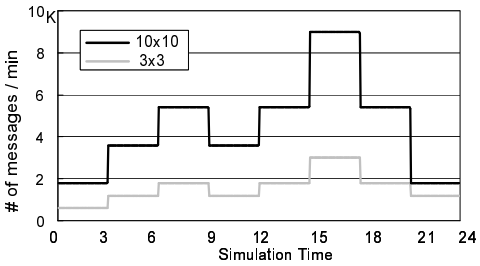

Fig. 37. Workload (10x10 and 3x3)

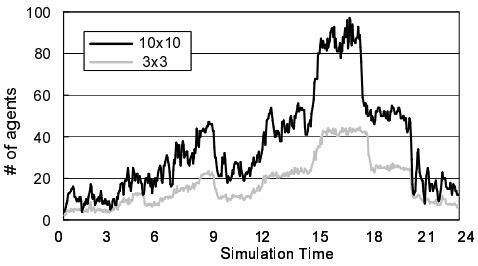

Fig. 38. Agent Population

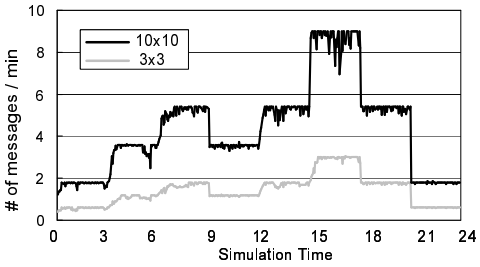

Fig. 39. Agent Throughput

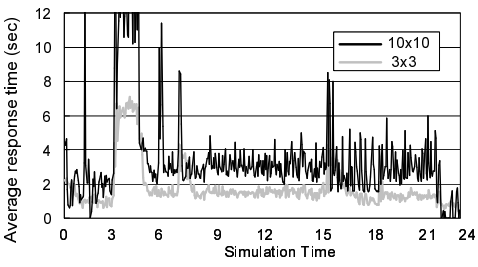

Fig. 40. Average Response Time for the User

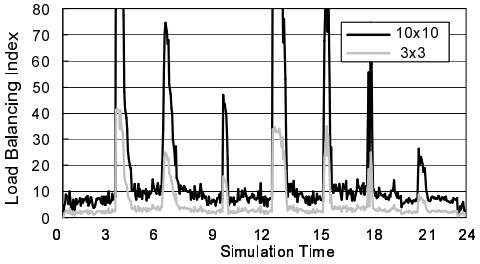

Fig. 41. Load Balancing Index

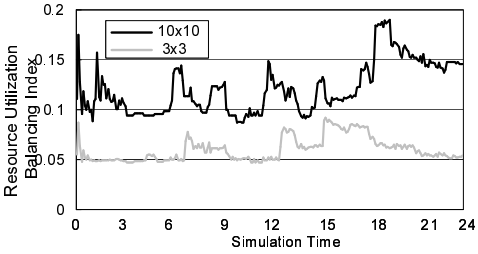

Fig. 42. Resource Utilization Balancing Index

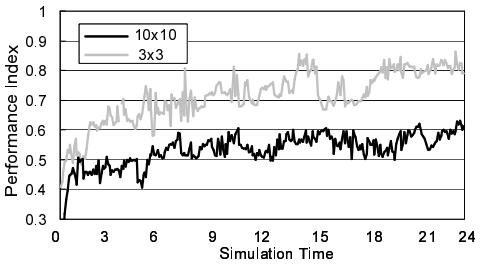

Fig. 43. Average Performance Index of Agents

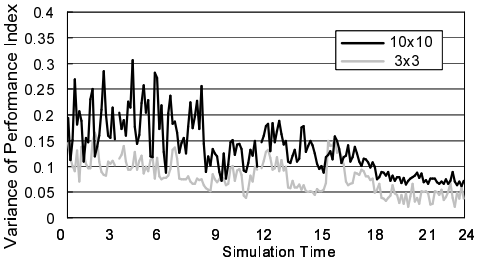

Fig. 44. Variance of Performance Index

time, throughput, and network lifetime.

In this simulation study, the simulated network is assumed as the Internet data center containing two server clusters (of a 7x7 grid each) as described in Figure 45. In the middle of simulation, the link between cluster 1 and a switch is broken at 15:30. Thus, the agents in cluster 1 are no longer accessible for the user. Then, it takes 90 min to be recovered (at 17:00). Figure 46 shows how agents adapt their population against link failures in each cluster 1 and 2. The number of agents in cluster 1 dramatically decreases at 15:30 because

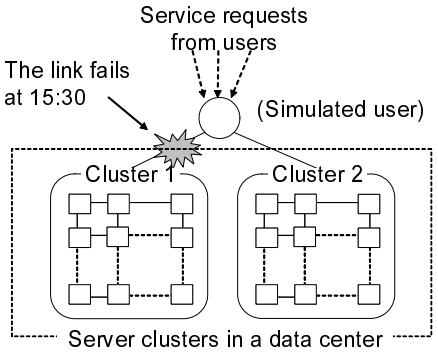

Fig. 45. Simulated Network

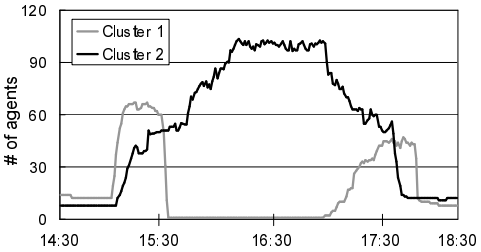

Fig. 46. Agent Population

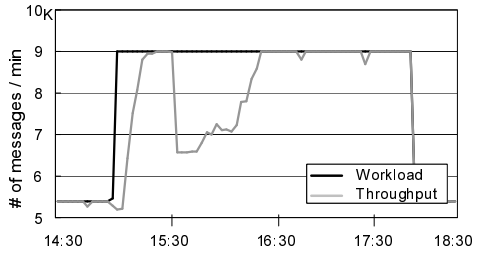

Fig. 47. Agent Throughput

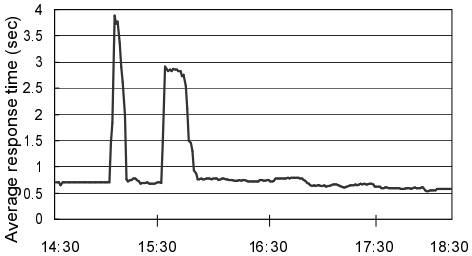

Fig. 48. Average Response Time for the User

the agents cannot get any service requests from the user and lose their energy. However, agents in cluster 2 start replicate themselves to process more service requests instead of the agents in cluster 1. When the link failure is fixed at 17:00, agents in cluster 1 (assume that one agent is manually invoked after cluster 1 is recovered) increase their population again.

Figure 47 shows how agents improve throughput against link failure. When the link failure occurs, the throughput will drop because the agents in cluster 1 are dying. However, the agents quickly improve their throughput even before cluster 1 is recovered by replicating them in cluster 2. Figure 48 shows how agents reduce the response time for the user against the link failure. At 15:30, the response time increases because not enough agents are running in the network to process all service requests. However, by increasing agent population in cluster 2, agents successfully reduce their response time for the user before cluster 1 is recovered. Agents autonomously and dynamically adapt their population to keep high throughput and low response time even during the link failure (15:30 to 17:00).

5. RELATED WORK

This paper describes a set of extensions to the authors' prior work [Lee and Suzuki 2005; 2006; 2007]. A preliminary design of iNet was reported in [Lee and Suzuki 2005; 2006]; however, it investigated no evolutionary mechanisms. While the iNet evolutionary mechanism was reported in [Lee and Suzuki 2007], this paper describes an enhanced version of the mechanism and reveals extended simulation studies on agent adaptability as well as new simulation results on scalability, survivability and self-organization.

Artificial immune systems have been proposed and used in various application domains such as anomaly detection in network hosts [Gonzalez and Dasgupta 2003], pattern recognition in images [de Castro and Timmis 2002], peer-to-peer content discovery [Ganguly

et al. 2004] and product recommendation [Chen and Aickelin 2004]. The negative selection process is applied in [Gonzalez and Dasgupta 2003]. The B-cell activation responses are applied in [de Castro and Timmis 2002; Ganguly et al. 2004; Chen and Aickelin 2004]. iNet applies both T-cell and B-cell activation responses so that they complement each other. To the best of the authors' knowledge, this work is the first attempt to apply an artificial immune system to the area of self-organizing and evolvable network applications.

The BEYOND architecture is similar to the Bio-Networking Architecture (BNA) [Nakano and Suda 2005] in that both have biologically-inspired agents evolve and adapt to dynamic network environments in a decentralized manner. However, they employ different adaptation mechanisms for agents. BEYOND investigates an artificial immune system (i.e., iNet) while BNA investigates a traditional genetic algorithm (GA). BNA does not have a mechanism equivalent to the iNet EE facility; thus, agents periodically invoke behaviors even when they have adapted to the current environment conditions. This can result in wasting resources caused by unnecessary behavior invocations. In BEYOND, the EE facility examines whether each agent adapts well to the current environment conditions, and activates the BS facility only when it does not. As discussed in Section 4, the EE facility stabilizes agent performance by avoiding unnecessary resource consumption and execution overhead. Moreover, a GA used in BNA employs a fitness function to rank agents in mating partner selection. The function contains a threshold value as well as a weight value for each objective. Agent designers need to manually configure these values through trial-and-errors. In iNet, no parameters exist for ranking agents because of a domination ranking mechanism. iNet requires much less configuration costs for agent designers.

Bionets allows each application to encode its information (e.g., its location and user profiles) as a set of genes and evolve it with a GA in order to improve user satisfaction [Carreras et al. 2006]. Similar to BNA, this GA uses a fitness function to rank agents, and the function contains a threshold value as well as a weight value for each objective. Therefore, agent designers incur manual configuration of the fitness function. iNet requires much less configuration costs for agent designers because no parameters exist for ranking agents.

Organic Grid is a decentralized task scheduling mechanism for grid applications [Chakravarti et al. 2005]. A computational task is assigned to a set of mobile agents, and they autonomously find and migrate to the hosts that provide enough resources to execute a subtask. iNet is similar to Organic Grid in that both allow agents to adapt their locations. However, iNet-enabled agents can perform many other types of adaptation as well such as the adaptation of response time, throughput and workload distribution. iNet also considers scalability and survivability, which are beyond of Organic Grid's scope.

In traditional rule-based adaptation mechanisms [Bouchenak et al. 2006; Hagimont et al. 2006; Terfloth et al. 2006; Bugajska and Schultz 2002], application developers need to anticipate all possible environment conditions and manually predefine a set of feasible adaptation rules against them. When the number of environment conditions grows, the number of rules often increases dramatically. Thus, manual configuration of rules can be complicated and error-prone. In contrast, iNet requires no manual configurations of antibodies (rules); it allows agents to autonomously evolve and tune their antibodies.

There are several structural similarities between the iNet BS facility and neural networks. Each neural network consists of neurons, which are connected with each other through synapses [Stergiou and Siganos 1996; Lee 2003]. The iNet BS facility consists of antibodies connected with each other with stimulation/suppression relationships. iNet

considers each antibody's concentration (i.e., an internal state of an antibody), while most neural networks do not consider each neuron's internal state. When iNet receives an input (i.e., a set of environment conditions), it selects an antibody as an output based on the concentration of each antibody. It memorizes the output as the selected antibody's concentration and uses the memory to efficiently select an antibody when it encounters the same input in the near future. Most neural networks do not provide this memory function.

6. CONCLUSION

This paper describes and evaluates an immunologically-inspired adaptation mechanism, called iNet, which allows network applications to be autonomous, scalable against workload volume and network size, adaptive to dynamic and heterogeneous network environments and survivable against link failures in the network. Inspired by the mechanisms behind how the immune system works, iNet allows each application component (agent) to autonomously sense its surrounding environment conditions and adaptively invoke a behavior suitable for the conditions. A configuration of antibodies is encoded as a gene, and antibodies evolve via genetic operations such as mutation and crossover.

Several extensions to iNet are planned. For example, the EE facility will be enhanced to implement evolutionary process and improve the quality of self/non-self classification. An extended set of empirical experiments is also planned to deploy and evaluate iNet on a large-scale experimental networks such as PlanetLab (www.planet-lab.org).

REFERENCES

Albert, R., Jeong, H., and Barabasi, A. 2001. Error and attack tolerance of complex networks. *Nature 406*, 378–382.

Berek, C. 2005. Somatic hypermutation and b-cell receptor selection as regulators of the immune response. *Transfusion Medicine and Hemotherapy 32,* 6, 333–338.

Bouchenak, S., Palma, N. D., Hagimont, D., and Taton, C. 2006. Autonomic management of clustered applications. In *Proc. of IEEE Int'l Conference on Cluster Computing*.

Bugajska, M. D. and Schultz, A. C. 2002. Coevolution of form and function in the design of micro air vehicles. In *Proc. of NASA/DoD Conference on Evolvable Hardware*.

Cabri, G., Leonardi, L., and Zambonelli, F. 2000. Mobile-agent coordination models for internet applications. *IEEE Computer 33,* 2, 82–89.

Camazin, S., Deneubourg, J. L., Franks, N. R., Sneyd, J., Theraula, G., and Bonabeau, E. 2003. *Self Organization in Biological Systems*. Princeton University Press.

Carreras, I., Chlamtac, I., Pellegrini, F. D., and Miorandi, D. 2006. Bionets: Bio-inspired networking for pervasive communication environments. *IEEE Trans. on Vehicular Technology 56,* 1, 218–229.

Chakravarti, A., Baumgartner, G., and Lauria, M. 2005. The organic grid: Self-organizing computation on a peer-to-peer network. *IEEE Trans. on Systems, Man, and Cybernetics 35,* 3, 373–384.

Chase, J., Anderson, D., Thakar, P., Vahdat, A., and Doyle, R. 2001. Managing energy and server resources in hosting centers. In *Proc. of the Eighteenth ACM Symposium on Operating Systems Principles*.

Chen, Q. and Aickelin, U. 2004. Movie recommendation systems using an artificial immune system. In *Proc. of the Sixth Int'l Conference in Adaptive Computing in Design and Manufacture*.

de Castro, L. N. and Timmis, J. I. 2002. Artificial immune systems: A novel paradigm to pattern recognition. In *Artificial Neural Networks in Pattern Recognition*, M. Corchado, L. Alonso, and C. Fyfe, Eds. Univ. of Paisley.

Deb, K. 2001. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley.

Dini, P., Gentzsch, W., Potts, M., Clemm, A., Yousif, M., and Polze, A. 2004. Internet, grid, self-adaptability and beyond: Are we ready? In *Proc. of IEEE Int'l Workshop on Self-Adaptable and Autonomic Comp. Sys.*

Ganguly, N., Canright, G., and Deutsch, A. 2004. Design of an efficient search algorithm for p2p networks using concepts from natural immune systems. In *Proc. of Int'l Conf. on Parallel Problem Solving from Nature*.

Gershenson, C. and Heylighen, F. 2003. When can we call a system self-organizing? In *Proc. of the Seventh European Conference on Artificial Life*.

Gonzalez, F. A. and Dasgupta, D. 2003. Anomaly detection using real-valued negative selection. *Genetic Programming and Evolvable Machines 4,* 4, 383–404.

Hagimont, D., Bouchenak, S., Palma, N., and Taton, C. 2006. Self-sizing of clustered databases. In *Proc. of the Seventh Int'l Symposium on World of Wireless, Mobile and Multimedia Networks*.

Jerne, N. K. 1984. Idiotypic networks and other preconceived ideas. *Immunological Review 79*, 5–24.

Lee, C. and Suzuki, J. 2005. Autonomic adaptation of network applications with the inet artificial immune system. In *Proc. of the Fourh IASTED Int'l Conf. on Communications, Internet and Information Technology*.

Lee, C. and Suzuki, J. 2006. Biologically-inspired design of autonomous and adaptive grid services. In *Proc. of the Second IEEE Int'l Conference on Autonomic and Autonomous Systems*.

Lee, C. and Suzuki, J. 2007. An immunologically-inspired adaptation mechanism for evolvable network applications. In *Proc. of the Fourth IEEE Consumer Communications and Networking Conference*.

Lee, M. 2003. Evolution of behaviors in autonomous robot using artificial neural network and genetic algorithm. *Information Sciences 155,* 1, 43–60.

Minar, N., Kramer, K. H., and Maes, P. 1999. Cooperating mobile agents for dynamic network routing. In *Software Agents for Future Communications Systems*, A. Hayzelden and J. Bigham, Eds. Springer.

Mitchell, T. 1997. *Machine Learning*. McGraw-Hill.

Nakano, T. and Suda, T. 2005. Self-organizing network services with evolutionary adaptation. *IEEE Trans. on Neural Networks 16,* 5, 1269–78.

Ranjan, S., Rolia, J., Knightly, E., and Fu, H. 2002. Qos-driven server migration for internet data centers. In *Proc. of the Tenth IEEE Int'l Workshop on Quality of Service*.

Rolia, J., Singhal, S., and Friedrich, R. 2000. Adaptive internet data centers. In *Proc. of Int'l Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet*.

Stergiou, C. and Siganos, D. 1996. Neural network. *Surveys and Presentations in Info. Systems Engineering 4*.

Sterritt, R. and Bustard, D. 2003. Towards an autonomic computing environment. In *Proc. of the Fourteenth IEEE Int'l Workshop on Database and Expert Systems Applications*.

Terfloth, K., Wittenburg, G., and Schiller, J. 2006. Facts - a rule-based middleware architecture for wireless sensor networks. In *Proc. of the First IEEE Int'l Conf. on Communication System Software and Middleware*.

Vasilakos, A., Parashar, M., S.Karnouskos, and W.Pedrycz. 2008. *Autonomic Communication*. Springer.

A. APPENDIX

The BEYOND simulator allows agent designers to visually configure iNet and run their agents on arbitrary network topologies (Figures 49 and 50).

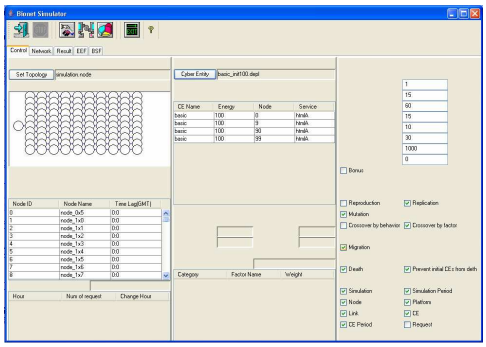

Fig. 49. The BEYOND Simulator (1). A GUI to Configure a Simulated Network.

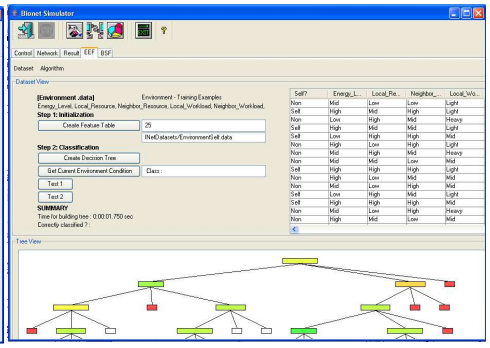

Fig. 50. The BEYOND Simulator (2). A GUI to Configure the iNet EE facility.