An Autonomic Adaptation Mechanism for Decentralized Grid Applications

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

Abstract

This paper describes and empirically evaluates a biologically-inspired adaptation mechanism that allows grid network services to autonomously adapt to dynamic environment changes in the network. Based on the observation that the immune system has elegantly achieved autonomous adaptation, the proposed mechanism, the iNet artificial immune system, is designed after the mechanisms behind how the immune system detects antigens (e.g. viruses) and specifically reacts to them. iNet models a set of environment conditions (e.g. network traffic and resource availability) as an antigen and a behavior of grid services (e.g. migration and replication) as an antibody. iNet allows each grid service to autonomously sense its surrounding environment conditions (i.e. an antigen) to evaluate whether it adapts well to the sensed conditions, and if it does not, adaptively perform a behavior (i.e. an antibody) suitable for the sensed conditions. Empirical evaluation results show that iNet works efficiently in acceptable degree of accuracy and makes grid services adaptive to dynamic network environment.

1. Introduction

Grid systems are expected to be more *autonomous* and *adaptive* to dynamic changes in the network (e.g. changes in network traffic and resource availability) in order to improve user experience, expand application's operational longevity and reduce maintenance cost [1, 2, 3]. As inspiration for a new design paradigm for grid applications, we observe various biological systems have already achieved autonomy and adaptability. We believe if grid applications are designed after certain biological concepts and mechanisms, they may be able to increase their autonomy and adaptability.

The NetSphere architecture applies key biological concepts and mechanisms to design grid applications. A grid application is modeled as a decentralized group of autonomous 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 migration, communication, replication, energy exchange and death.

This paper addresses autonomous adaptability of grid applications (i.e. agents). The proposed adaptation mechanism, iNet, is designed after the mechanisms behind how the immune system detects antigens (e.g. viruses) and produces specific antibodies to kill them. iNet models a set of environment conditions (e.g. network traffic) as an antigen and a behavior of agents as an antibody. iNet allows each agent to autonomously sense its local environment conditions (i.e. an antigen) to evaluate whether it adapts well to the sensed conditions, and if it does not, adaptively perform a behavior (i.e. an antibody) suitable for the conditions. For example, agents may invoke migration behavior for moving towards network hosts that accept a large number of user requests for their services. This leads to the adaptation of agent locations, and agents can reduce response time for users. Empirical evaluation results show iNet works efficiently in acceptable degree of accuracy and makes agents adaptive to dynamic network environment.

This paper is organized as follows. Section 2 overviews the design of agents in the NetSphere architecture. Section 3 describes the design of iNet. Section 4 shows empirical evaluation results. Sections 5 and 6 conclude with comparison with existing related work.

2. NetSphere architecture

In the NetSphere architecture, agents are designed based on the three principles described below [4].

Decentralization: Agents are decentralized. There are no central entities to control and coordinate agents (i.e. no directory servers and no resource managers). Decentralization allows grid services to be scalable and simple by avoiding performance bottleneck and any central coordination in deploying them [5, 6].

Autonomy: Agents are autonomous. Agents monitor their local network environments and autonomously behave and interact without any interventions from/to other agents, platforms and human users.

Adaptability: Agents are adaptive to dynamic environment conditions (e.g. user demands and resource

availability). Each agent contains iNet, which allows it to adaptively behave to the current environment.

Each agent is implemented as a Java object and runs on a NetSphere platform. The platform is also implemented in Java and runs atop a Java VM on a network host [4]. Each agent consists of *attributes*, *body* and *behaviors* [4]. Attributes carry descriptive information regarding the agent (e.g. agent ID). The body implements a service 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 non-service related actions that are inherent to all agents. Although NetSphere defines nine standard agent behaviors [4], this paper focuses on four of them.

Migration: Agents may move between platforms.

Energy exchange and storage: Agents may receive and store *energy* in exchange for providing services to other agents or users. Agents may also expend energy for services that they receive from other agents, and for resources available on a platform (e.g. memory space).

Communication: Agents may communicate with other agents for the purposes of, for example, requesting a service or exchanging energy.

Lifecycle regulation: Agents may regulate their lifecycles. They may make their copies (replication) in response to higher energy level. They also may die as a result of energy starvation. If energy expenditure of an agent is not balanced with the energy gain, the agent will not be able to pay for the resources it needs, i.e., it dies from lack of energy.

3. The iNet artificial immune system

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

3.1. Natural Immune System

The immune system is an adaptive defense mechanism to regulate the body against dynamic environment changes (e.g. antigen invasions). Through a number of interactions among various white blood cells (e.g. macrophages and lymphocytes) and molecules (e.g. antibodies), the immune system evokes two responses to antigens: *innate* and *adaptive* immune response.

In the innate immune response, the immune system performs *self/non-self discrimination* to detect antigens. This response is initiated by macrophages and T-cells, a type of lymphocytes. Macrophages move around the body to ingest antigens and present them to T-cells so that T-cells can recognize them. T-cells are produced in thymus and trained through the *negative selection process*. In this process, thymus removes T-

cells that react with the body's own (self) cells. The remaining T-cells are used as detectors to identify non-self cells (i.e. antigens). When T-cells detect non-self cells, they secrete chemical signals to activate the second immune response: adaptive immune response.

In the adaptive immune response, the immune system produces antibodies that specifically react and kill an antigen identified by T-cells. Antibodies form a network structure and communicate with each other [7]. This network is formed with stimulation and suppression relationships among antibodies. Thus, the adaptive 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 helps to keep the quantitative balance of antibodies. Through the stimulation and suppression interactions, the population of specific antibodies rapidly increases following the recognition of an antigen and, after eliminating the antigen, decreases again. Performed based on this self-regulation mechanism, the adaptive immune response is an emergent product from many interactions among antibodies.

3.2. Design and implementation of iNet

The iNet artificial immune system consists of the *environment evaluation* (EE) facility and *behavior selection* (BS) facility (Figure 1) corresponding to the innate and adaptive immune response, respectively. EE allows each agent to continuously sense a set of current environment conditions as an antigen and examine whether it is 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 EE detects a non-self antigen, EE activates BS (Figure 1). BS allows each agent to choose a behavior as an antibody that specifically matches with the detected non-self antigen.

3.2.1. Environment evaluation facility (EE)

EE performs two steps: initialization and self/non-self classification (Figure 2). The initialization step produces detectors that identify self and non-self antigens (i.e. environment conditions). In iNet, an antigen
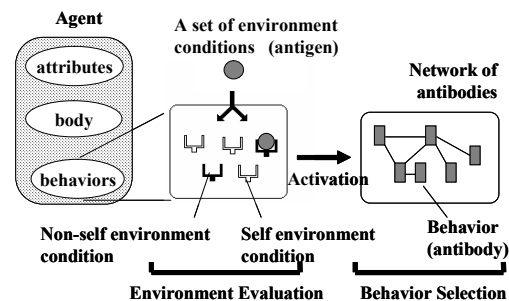

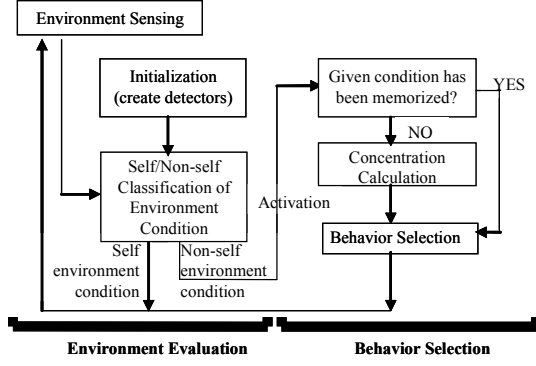**Figure 1. Organization of the iNet artificial immune system**

Figure 2. iNet Adaptation Process

(i.e. a set of environment conditions) is implemented as a *feature vector*. Each feature vector (*X*) consists of a set of features (*F*) and a class value (*C*). *F* contains a series of environment conditions. If an agent senses agent population on a local platform, resource utilization on a local platform and workload (the number of user requests) on a local platform, a feature vector may be represented such as X_{curent}=*((Low: Agent population, Low: Resource utilization, Heavy: Workload), C)*. *C* indicates whether a given antigen (i.e. a set of environment conditions) is self (0) or non-self (1).

To evaluate whether an antigen (i.e. feature vector) is self or non-self, the initialization step produces detectors that identify them (Figure 2). This step is designed after the negative selection process in the immune system. In the initialization step, EE first generates feature vectors randomly, and separates them into self detectors, which closely match with self antigens (feature vectors), and non-self detectors[1], which do not closely match with self antigens (feature vectors). This separation is performed via vector matching between randomly generated feature vectors and self antigens (feature vectors) that human users supply (Figure 3). Currently, EE uses the Euclidean vector matching algorithm. After vector matching, both self and non-self detectors are stored in a feature table (Figure 3)[2]. The second step in EE performs self/non-self classification of environment conditions (Figure 2). It uses the detectors in a feature table to classify the current environment conditions into self or non-self. The self/non-self classification step is performed with a decision tree built from detectors in a feature table. Figure 4 shows an example decision tree. EE starts to examine a set of given current environment conditions, $X_{current}$, at the root of the decision tree. Each node in the tree specifies which feature is considered. Based on the value of the specified feature in $X_{current}$, EE follows
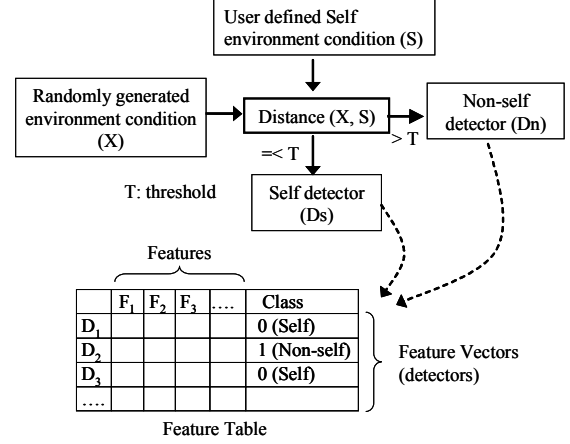
---

[1] Non-self detectors in iNet are equivalent to T-cells in the immune system.
[2] The immune system removes non-self detectors through negative selection process. However, in iNet, both self and non-self detectors are kept in a feature table to perform self/non-self classification.

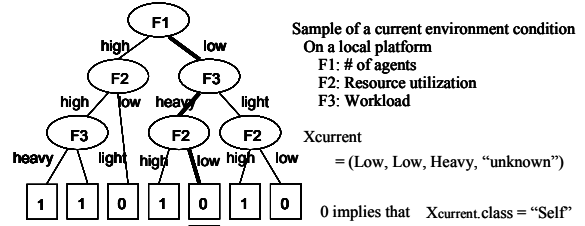**Figure 3. Initialization Step in EE**

Figure 4. An Example Decision Tree

down along the branch indicating the value. This process is repeated until EE reaches at the leaf of tree which notices the class value of $X_{current}$. Once EE detects a non-self antigen, it activates BS immediately.

The reasons for using decision tree as a classifier are ease of implementation and algorithmic efficiency. Since a decision tree is easy to understand and implement, iNet can maintain a lower barrier for developers to design adaptive grid applications. Also, a decision tree performs classification much faster than other algorithms such as clustering, support vector machine and Markov model algorithms [8, 9]. The efficiency of classification is one of the most important requirements in iNet because each agent periodically senses and classifies its surrounding environment conditions.

3.2.2. Behavior selection facility (BS)

Once EE classifies the current environment conditions as a non-self antigen, it activates BS. BS selects an antibody (i.e. agent's behavior) suitable for the detected non-self antigen (i.e. environment conditions). Each antibody is structured as shown in Figure 5. It consists of *Paratope*, precondition under which it is selected (one of environment conditions), *Behavior ID*, one of agent behaviors, and *Idiotope*, relationships to other antibodies (one or more links). Antibodies are linked with each other using stimulation and suppres-

Antibody

Precondition under which this antibody is selected	Agent Behavior ID	Relationships to other antibodies (behaviors)
Paratope	Behavior	Idiotope

Figure 5. Antibody Structure

Figure 6. A Generalized Network of Antibodies
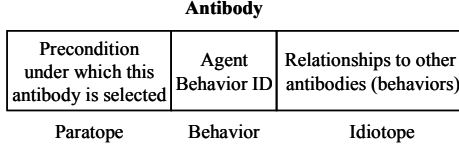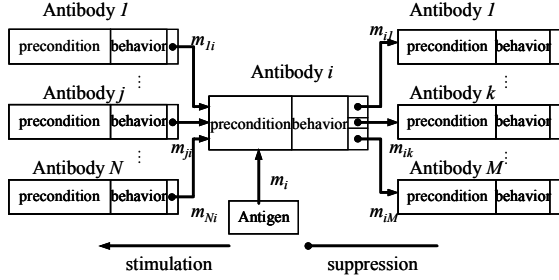

Figure 7. An Example Network of Antibodies

sion relationships (see Section 3.1). Each antibody has its own concentration value corresponding to the number of the antibody. The value is used to prioritize antibodies (behaviors) in behavior selection. BS 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 antigens (behaviors), stimulation relationships between them contribute to increase their concentration values, and suppression relationships contribute to decrease it. Each relationship has its own strength (affinity), which indicates the degree of stimulation or suppression.

Figure 6 shows a generalized 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. 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.

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

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

In the equation (1), the first and second terms in a big bracket denote the stimulation and suppression from other antibodies. The affinity values between antibodies (i.e. m_{ji} and m_{ik}) are positive between 0 and 1. m_i is 1 when antibody i is stimulated directly by an antigen, otherwise 0. k denotes the dissipation factor representing the natural death of an antibody. This value is 0.1. The initial concentration value for ever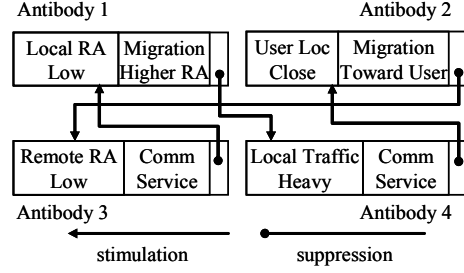y antibody, $a_i(0)$, is 0.01. The equation (2) is a sigmoid function used to squash the $A_i(t)$ value between 0 and 1.

Every antibody's concentration is calculated 200 times repeatedly. This repeat count is obtained from a previous simulation experience [10, 11]. If no antibody exceeds a predefined threshold (0.7) during the 200 calculation steps, the antibody whose concentration value is the highest is selected (i.e. winner-tales-all selection). If one or more antibodies' concentration values exceed the threshold, an antibody is selected based on the probability proportional to the current concentrations (i.e. roulette-wheel selection).

Figure 7 shows an example network of antibodies. It contains four antibodies, which represent communication behavior and migration behavior with two different policies. Antibody 1 represents the migration behavior invoked when resource availability is low on the local platform. Antibody 1 suppresses Antibody 3 when it is stimulated (i.e. when resource availability is low on the local platform). Now, suppose that a (non-self) antigen indicates (1) resource availability is low on the local platform, (2) network traffic is low on the local platform and (3) user location is close. This antigen stimulates Antibodies 1, 2 and 4 simultaneously. Their population increases, and it is likely that Antibody 2's concentration value becomes highest because Antibody 2 suppresses Antibody 4, which in turn suppresses Antibody 1. As a result, Antibody 2 (i.e. migration behavior) would be selected.

4. Empirical evaluation results

This section shows empirical evaluation results to examine the efficiency and accuracy of iNet and the adaptability of grid services (agents) developed with iNet. The efficiency and accuracy of iNet are evaluated with a Java 2 standard edition JVM on a Windows XP PC with a 2.5GHz Intel Celeron CPU and 1GB memory. The adaptability of agents is evaluated with a maximum of nine Windows XP PCs, each PC hosts a NetSphere platform on a Java2 standard edition JVM with 2.0GHz Intel Celeron CPU and 512MB memory. Those PCs are connected through 100Mbps Ethernet.

4.1. Efficiency and accuracy of iNet

The overhead of EE (T_{EE}) includes the initialization time (T_{init}) and classification time ($T_{classify}$). T_{init} consists of the time (t1) to generate self/non-self detectors, and the time (t2) to build a decision tree (i.e. $T_{EE} = T_{init} + T_{classify}$). Table I shows that T_{init} grows in proportion to the values of N (the size of a feature table), A (the number of features in each feature vector) and classification accuracy. For example, if there are four features in each feature vector and iNet is expected to achieve 90% classification accuracy, EE needs to have 90 detectors in a feature table and its initialization time is approximately 8.6 seconds. Please note that agents (grid applications) do not incur the initialization overhead at runtime because EE performs initialization before running agents.

Table II shows $T_{classify}$; how long it takes for EE to classify an antigen (a set of environment conditions) into self or non-self. Agents (grid applications) incur classification overhead at runtime. However, the overhead is small enough and acceptable in most agents (applications).

Figure 8 shows how EE is accurate to classify an antigen (a set of environment conditions) into self or non-self. Classification accuracy increases as the values of N and A grow. In general, EE can achieve 90% classification accuracy with small overhead depicted in Table II. There is a trade-off between efficiency (T_{init} and $T_{classify}$) and classification accuracy. Application developers need to determine the value of N, based on
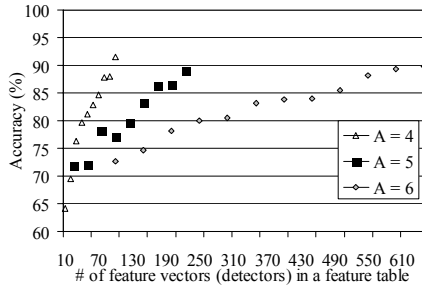

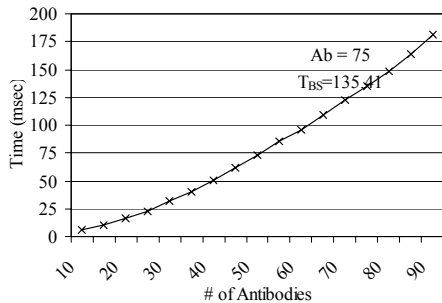**Figure 8. Classification Accuracy when A=4, 5, and 6**

**Figure 9. T_{BS}, Overhead of BS facility,
i.e. time for selecting a behavior**

**Table I. Initialization Time (T_{init})
with A=3, DT could not achieve 90% classification accuracy.
N: recommended size of a feature table**

		# of features (A)	3	4	5
80%	Tinit (sec)	t1	0.0015	0.0016	0.0034
		t2	2.21	3.913	9.874
		Total	2.212	3.915	9.877
	Size of feature table (N)		**20**	**40**	**75**
85%	Tinit (sec)	t1	0.0015	0.0031	0.0094
		t2	2.767	7.121	16.101
		Total	2.769	7.124	16.111
	Size of feature table (N)		**25**	**70**	**150**
90%	Tinit (sec)	t1	-	0.0034	0.0141
		t2	-	8.748	18.528
		Total	-	8.751	18.542
	Size of feature table (N)		-	**90**	**225**

Table II. $T_{classify}$: Overhead of Classification

# of features (A)	3	4	5	6	..	10
Tclassify (msec)	1.5	3.0	3.0	3.0		4.5

the results shown in Tables I and II, depending on the requirements of their applications.

The overhead of BS (T_{BS}) represents the time to select an antibody (behavior) suitable for a given antigen (a set of environment conditions). In BS, the number of antibodies (Ab) is determined by the number of features in each feature vector (A), the number of distinct values of each feature (V) and the number of behaviors that each agent supports (B). According to the antibody structure (Figure 5), it will contain a value of one of features and one of behaviors. Therefore, the number of all possible types of antibodies would be calculated as $Ab = (A*V)*B$. As shown in Figure 9, the overhead of BS exponentially increases as Ab grows.

Compared with T_{init} and T_{BS}, $T_{classify}$ is very small. For example, when A=5, EE requires T_{init}=9.877sec at least for the initialization (Table I), and BS does T_{BS}=135.41msec for behavior selection with a network of 75 antibodies (e.g. Ab=(5*3)*5) (Figure 9). T_{BS} will be skipped if the classification spends $T_{classify}$=3msec (Table II) and says a current environment condition is "Self". EE works well to keep the overhead of iNet lightweight by eliminating T_{BS} when necessary.

4.2. Autonomous adaptability of agents

In this experiment, the testbed network deploys four NetSphere platforms on four PCs (i.e. a platform on each PC) at the beginning of an experiment (0:00). Three hours later (at 3:00), five more PCs are added to the testbed network, and nine NetSphere platforms work on nine PCs from 3:00 to 7:00 (Figure 10). The platforms (and PCs) are connected with each other based on a grid (2x2 or 3x3) topology (Figure 10).

At the beginning of an experiment, a single web service agent is randomly deployed on a platform. Each agent contains iNet that is configured with four behaviors (migration, communication, replication and death) and seven environment conditions (resource availability on the local platform and a one-hop away remote platform, workload on the local platform and a one-hop away platform, energy level, user location, the number of agents running on the local platform). A workload generator simulates a user (Figure 10). It generates HTTP request messages and randomly sends them to agents. It keeps track of the locations of agents. When an agent migrates, the agent notifies its new location to the workload generator[3]. The workload generator pays energy to an agent when it receives an HTTP response message from the agent.

Figure 11 shows how agents autonomously adapt their throughput to workload changes by adjusting their population and locations. Figure 11 (1) shows the workload for agents (i.e. how many HTTP request messages a user generates and sends to agents). The workload keeps around 600 messages/min from 0:00 to 1:00, spikes to approximately 12,000 messages/min at 1:20, and drops to 600 messages/min at 5:20. Figure 11 (2) shows how agents change their population against workload changes. As agents perform replication behavior in response to enough energy gain from a user, their population gradually grows. This allows agents to process more HTTP request messages from a user. Also, agents migrate to platforms where resource availability is higher. This prevents agents to excessively crowd on a single platform, and contributes to evenly distribute agents over platforms. As shown in Figure 11 (3), agents autonomously increase their throughput as they perform replication and migration behaviors. The agent throughput represents how many HTTP response messages agents send back to a user.

From 2:20 to 3:00, the agent throughput does not increase although a user generates high workload. The agent population does not increase either. In this period, agents cannot replicate themselves because a number of agents (approx 35 agents) work on four platforms and resource availability is very low on the platforms. When five more platforms (PCs) enter to the testbed network at 3:00, agents migrate to the new platforms, where resource availability is higher, and replicate themselves on the new platforms. After that, the agent population and throughput increase again to process HTTP request messages from a user (see Figure 11).

When the workload drops at 5:00, many agents cannot gain enough energy to keep living because they

cannot receive enough HTTP request messages from a user. Due to energy starvation, some of agents die, resulting in a drop of agent population. This prevents unnecessary agents from staying in the network and consuming resources.

5. Related work

Artificial immune systems have been proposed and used in various application domains such as anomaly detection [12] and pattern recognition [13]. [12] mainly focuses on the generation of detectors for self/non-self classification and improves the negative selection process of the artificial immune system. [13] focuses on the
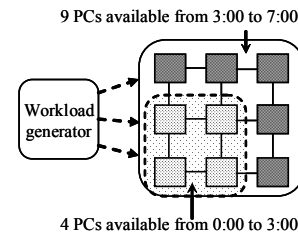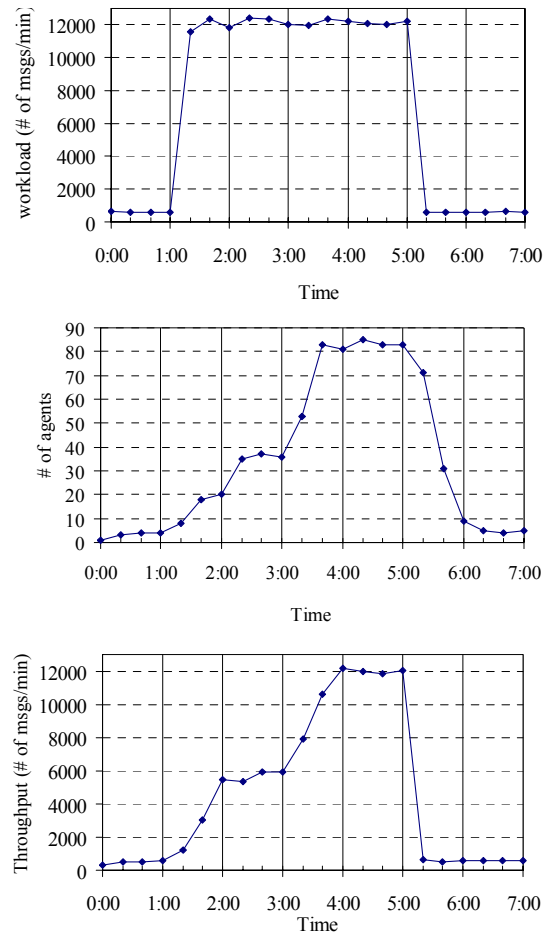

Figure 10. Testbed Architecture

Figure 11. (1) generated workload, (2) the number of agents, (3) throughput of agents.

---

[3] In principle, agents are decentralized as described in Section 2. However, for simplicity, the workload generator in this measurement plays a role of a directory that maintains the locations of agents.

accuracy for the matchmaking of an antigen and antibody. On the other hand, this paper proposes an artificial immune system to improve autonomous adaptability in grid network services. To the best of our knowledge, this work is the first attempt to apply an artificial immune system into the autonomic adaptation domain.

Similar work has been proposed in Organic Grid [14] project. [14] attempts to the decentralized task scheduling for large-scale computation on grid environment over the Internet. Similar to iNet, mobile agents autonomously executes their services (e.g. computing subtasks) on the platform embedded in each host and perform their replication behavior to achieve their objectives (e.g. compute as fast as possible). Yet, iNet focuses on the grid service adaptation. Agents consider more behaviors for adaptation, and through those various adaptation decisions, the high adaptability of grid network services (i.e. agents) is achieved.

There are several research efforts that allow network systems to adapt to application and end-user requirements with a technique of runtime component replacement. For example, [15, 16] can dynamically replace running components (e.g. byte code) with others. It is possible to change the functional aspect of components. iNet does not modify the body (functional part) of agents at runtime; it focuses on adapting the behavior of agents (non-functional part) in order to minimize runtime adaptation overhead. The overhead should be cheap because iNet assumes each agent often performs its behavior to keep adapting to dynamically changing environment while it should be considerably expensive to perform even a single runtime component replacement. Another difference is that [15, 16] assumes a centralized network architecture where a centralized server collects environment conditions from each component to make replacement decisions. In contrast, iNet assumes a decentralized network architecture where each agent monitors its surrounding environment and makes adaptation decisions.

6. Concluding remarks

This paper describes and empirically evaluates a biologically-inspired mechanism that allows grid services to autonomously adapt to dynamic changes in the network. The proposed adaptation mechanism, called the iNet artificial immune system, allows each grid service to autonomously sense its local environment conditions to evaluate whether it adapts well to the conditions, and if it does not, adaptively perform a behavior suitable for the conditions. Empirical evaluation results show that iNet works efficiently with high degree of accuracy and makes grid services adaptive.

7. 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 Int'l Workshop on Self-Adaptable and Autonomic Computing Systems*, Aug. 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*, Mar 2001.

[3] R. Sterritt and D. Bustard, "Towards an Autonomic Computing Environment," In *Proc. of 14th IEEE International Workshop on Database and Expert Systems Applications*, Sep 2003.

[4] J. Suzuki and T. Suda, "A Middleware Platform for a Biologically-inspired Network Architecture Supporting Autonomous and Adaptive Applications," *IEEE Journal on Selected Areas in Communications*, vol 23, 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," In *IEEE Computer*, Feb 2000.

[7] N.K.Jerne, "Idiotypic Networks and Other Preconceived Ideas," In *Immunological Review,* vol. 79, 1984.

[8] P. Berkhin, "Survey of Clustering Data Mining Techniques," Accrue Software, Inc., San Jose, CA, 2002.

[9] T. Mitchell, *Machine Learning*, McGraw-Hill, 1997.

[10] J. Suzuki and Y. Yamamoto, "iNet: An Extensible Framework for Simulating Immune Network," In *Proc. of IEEE International Conference on Systems, Man, and Cybernetics,* October 2000.

[11] J. Suzuki and T. Suda, "Adaptive Behavior Selection of Autonomous Objects in the Bio-Networking Architecture," In *Proc. of 1st Annual Symposium on Autonomous Intelligent Networks and Systems*, May 2002.

[12] F. A. González, D. Dasgupta, "Anomaly Detection Using Real-Valued Negative Selection," *Genetic Programming and Evolvable Machines,* 4(4), 2003.

[13] L. N. de Castro, J. I. Timmis, "Artificial Immune Systems: A Novel Paradigm to Pattern Recognition," In *Artificial Neural Networks in Pattern Recognition*, J. M. Corchado, L. Alonso, and C. Fyfe (eds.), SOCO-2002, University of Paisley, UK, pp. 67-84.

[14] A.J. Chakravarti, G. Baumgartner, M. Lauria, "The Organic Grid: Self-organizing Computational Biology on Desktop Grid," In *A Zomaya, Parallel Computing for Bioinformatics*, John Wiley & Sons, 2005.

[15] S. Cheng, D. Garlan, B. Schmerl, P. Steenkiste, and N. Hu, "Software Architecture-based Adaptation for Grid Computing," In *the 11th IEEE Conference on High Performance Distributed Computing,* July 2002.

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