社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－AL－102 (8)
2005／9／16

動的ネットワークにおける生態系パラダイムに基づく静的資源数制御

鈴木朋子†，泉泰介†，大下福仁†，角川裕次†，増澤利光†

近年，ファイルなどの静的資源をユーザ間で共有するアプリケーションが数多く存在している．このようなアプリケーションでは，資源の複製を生成しそれらをネットワーク中に分散配置することで，検索効率の向上が可能となる．しかし，ネットワーク中に存在する複製数が多いほど資源の検索時間は減少するが，必要となる記憶容量が増加する．従って，ネットワーク中の静的資源数をネットワークの規模に対して適切に設定することが重要である．本研究では，動的に変化するネットワークにおいて，静的資源数をノード数の一定割合に保つ問題について考察し，生態系パラダイムに基づいた手法を提案する．また，提案手法が静的資源数をノード数に対して所定の比に保つことをシミュレーションによって示す．

A Biologically Inspired Approach to Replica Control in Dynamic Networks

Tomoko Suzuki†, Taisuke Izumi†, Fukuhito Ooshita†, Hirotsugu Kakugawa†, Toshimitsu Masuzawa†

Resource replication is a crucial technique for improving system performance of distributed application with shared resources. A larger number of replicas require shorter time to reach a replica of the requested resource, but consume more storage of hosts. Therefore, it is indispensable to adjust the number of replicas appropriately for its application. This paper considers the problem for controlling the density of replicas adaptively in dynamic networks. The goal of the problem is to adjust the number of replicas to a constant fraction of the current network size. This paper proposes algorithm inspired by the single species population model, which is a well-known population ecology model. The simulation results show that the proposed algorithm realize self-adaptation of the replica density in dynamic networks.

1 Introduction

One of the most important advantages distributed applications inherently have is resource sharing. A well-known example is file sharing on peer-to-peer networks [7, 6]. In such applications, each resource is accessed frequently by a significant number of users distributed over the whole network.

For such applications with shared resources, resource replication is a crucial technique for improving system performance and availability: replicas of an original resource are distributed over the network so that each user can get the requested resource by accessing a nearby replica. Resource replication can reduce communication latency and consumption of network bandwidth, and can also improve availability of the resources even when some of the replicas are unavailable.

In systems using resource replication, generally, a larger number of replicas require shorter time to reach a replica of the requested resource, but consume more storage of hosts. Therefore, it is indispensable to adjust the number of replicas appropriately for its application. For example, resource searching protocol PWQS has tradeoff between the reach time and the number of replicas, and requires replicas of each resource in numbers proportional to the network size (i.e., the number of hosts) to attain good performance [9].

However, in dynamic networks such as peer-to-peer networks, the appropriate number of replicas for its application changes with time, since network size varies with time. In addition, it is unrealistic to assume that each node knows the network size and the number of replicas on the network. Therefore, adjustment of the number of replicas for dynamical change of network size is not an easy task and requires investigation.

Biological systems inherently have self-∗ properties , such as self-adaptation, self-configuration, self-optimization and self-healing, to realize environmental adaptation. Thus, several biologically-inspired approaches have succeeded in realizing highly adaptive distributed systems. Successful projects include Bio-Networking project[2] and Anthill project[1]. These projects adopt biologically-inspired approaches to provide highly adaptive platform for mobile-agent-based computing[4, 12]. In our precedence work, we also focus on biological system to control mobile agent population in dynamic networks[13]. Our algorithms are inspired by the well-known *the single species population model* and can adequately adjust the agent population in dynamic networks.

Contribution of this paper. In this paper, we first formulate *the replica density control problem* in dynamic networks, and present a biologically-

†大阪大学大学院情報科学研究科コンピュータサイエンス専攻
Department of Computer Science,Graduate School of Engineering Science, Osaka University

inspired solution for the problem. The replica density control problem requires to adapt the number of replicas to a given constant fraction of the current network size.

We propose a distributed solution for the problem using mobile agents. Mobile-agent-based distributed computing is one of the most promising paradigms to support autonomic computing in a large scale of distributed system with dynamics and diversity[10, 11]. Mobile agents are autonomous programs that can migrate from one node to another on the network, and traverse the distributed system to carry out a sophisticated task at each node.

To realize self-adaptation of the replica density, we borrow an idea from *the single species population model*, which is a well-known population ecology model. This model considers population of a single species in an environment such that individuals of the species can survive by consuming food supplied by the environment. The model is formulated by *the logistic equation* and shows that the population automatically converges to and stabilizes at some number depending on the amount of supplied food.

In the proposed algorithm, replicas of a resource are regarded as individuals of a single species, and agents created by nodes supply food for replicas. The algorithm try to adjust the number of replicas to a constant fraction of the network size by controlling the amount of food supplied by agents. The simulation results of the algorithm show that the proposed strategy can adequately adjust the replica density.

The rest of this paper is organized as follows. In Section 2, we present the model of distributed systems, and define the replica density control problem. In Sections 3 and 4, we propose the distributed solution for the problem and show its simulation results. Section 5 concludes the paper.

2 Preliminaries

2.1 System Models

Dynamic networks. In this paper, we consider *dynamic networks* such that its node set and its link set vary with time. To define dynamic networks, we introduce discrete time and assume that each time is denoted by a non-negative integer in a natural way: time 0 denotes the initial time, time 1 denotes the time immediately following time 0 and so on.

Formally, a dynamic network at time t is denoted by $N(t) = (V(t), E(t))$ where $V(t)$ and $E(t)$ are respectively the node set and the link set at time t. A link in $E(t)$ connects two distinct nodes in $V(t)$ and a link between nodes u and v is denoted by e_{uv} or e_{vu}. We also use the following notations to represent the numbers of nodes and edges at time t: $n(t) = |V(t)|$ and $e(t) = |E(t)|$.

Mobile agent systems. A *mobile agent* is an autonomous program that can migrate from one node to another on the network. In dynamic networks, agents on node $u \in V(t)$ at time t can start migrating to node $v \in V(t)$ only when link e_{uv} is contained in $E(t)$. The agent reaches v at time $t+\Delta$ only when the link e_{uv} remains existing during the period from t to $t+\Delta$, where Δ is an integer representing *migration delay* between the nodes. The agent migrating from u to v is removed from the network when the link e_{uv} disappears during the period from t to $t+\Delta$.

Each of nodes and agents has a *local clock* that runs at the same rate as the global time. However, we make no assumption on the local clock values: the difference between the local clock values in the system is unbounded.

An agent and a node can interact with each other by executing operations: agent p on node u can change its state and the state of u depending on the current states of p and u, and node u can change its state and the states of the agents residing on u depending on the current states of u and the agents. Besides the above operations, each agent can execute operations to create new agents and to kill itself and each node can also execute operations to create new agents.

When agents reside on a node, the agents and the node have operations they can execute. For execution semantics, we assume that the agents and the node execute their operations sequentially in an arbitrary order. We also assume that the time required to execute the operations can be ignored, that is, we consider all the operations are executed sequentially but at an instant time.

2.2 Replica Density Control

For an application with shared resource, resource replication is crucial technique for improving system performance. Resources are items shared by the nodes on the network; files, documents, and so on. Replicas are copy of an original resource. In such systems, generally, a larger number of replicas lead to better performance, but consume more storage of hosts. Thus, it is required to control the number of replicas appropriately for its application.

In this paper, we consider the *replica density control problem*. Each node has zero or more replicas. We consider a original resource as its replica. Each node v can make same replicas from a replica on node v and delete replicas on node v. The goal of the problem is to control the number of replicas of a resource so that the ratio between the number of replicas and the number of nodes (called *network*

size hereinafter) is kept to be a given constant. Let $r(t)$ be the number of replicas on the network $N(t)$ at time t: $r(t)$ is sum of the number of original resources and the number of its replicas. The problem is defined as follows.

Definition 2.1
The goal of the replica density control problem is to adjust the number of replica $r(t)$ at time t to satisfy the following equality for a given constant δ $(0 < \delta \leq 1)$.

$$r(t) = \delta \cdot n(t)$$

In this paper, we propose distributed solution for the replica density control problem. In the distributed solution, we assume that the constant δ is initially given to every node.

We consider distributed systems such that replicas are distributed over the networks and nodes can leave or join the networks. In such environment, it is obviously impossible to keep satisfying the above equation all the time. Thus, our goal is to propose distributed solution that realize quick convergence to and stability at the target number.

3 Replica Density Control Algorithm

In this section, we present a distributed solution for the replica density control problem. This algorithm is inspired by the single species population model (the logistic model), which is well-known in the field of the population ecology.

3.1 Single Species Population Model

In this subsection, we introduce the *single species population model* in the population ecology as the basis of our algorithm. This model considers an environment with a single species such that individuals of the species can survive by consuming food supplied by the environment. The model formulates the population growth of the species in the environment, and shows that the population (i.e., the number of individuals) in the environment automatically converges to and stabilizes at some number depending on the amount of food supplied by the environment.

We present more details of the single species population model. Each individual of the species periodically needs to take a specific amount of food to survive. That is, if an individual can take the specific amount of food then it can survive. Conversely, if an individual cannot take the specific amount of food then it dies. Moreover, in the case that an individual can take a sufficient amount of extra food, then it generates progeny. Consequently, the followings hold: The shortage of supplied food results in decrease in the population. Conversely, the excessive amount of food results in increase in the population.

The single species population model formulates the above phenomena as follows: Let $p(t)$ be the population at time t. The single species population model indicates that the *population growth rate* at time t is represented by the following nonlinear first-order differential equation known as the *logistic equation*[8]:

$$\frac{\Delta p(t)}{\Delta t} = p(t) \cdot g(t) = p(t)(k \cdot f_a(t) - k \cdot f \cdot p(t)),$$

where $f_a(t)$ is the amount of food supplied by the environment at time t, and f is the amount of food consumed by one individual to survive.

The *per capita growth rate* $g(t)$ at time t is represented by

$$g(t) = k(f_a(t) - f \cdot p(t)).$$

The expression $f_a(t) - f \cdot p(t)$ represents the difference between the amounts of supplied food and consumed food. When the supplied food exceeds the consumed food, $g(t)$ takes a positive value proportional to the difference, that is, the positive per capita growth rate $g(t)$ is proportional to the amount of the surplus food. A scarcity of the supplied food causes a negative value of $g(t)$ proportional to the difference, that is, the negative per capita growth rate $g(t)$ is proportional to the shortage of the supplied food.

The logistic equation has two equilibrium points of the population size $p(t)$: $p(t) = 0$ and $p(t) = f_a(t)/f$. That is, the population remains unchanged, when the population size is at the equilibrium points. The equilibrium point $p(t) = f_a(t)/f$ represents the maximum population that the environment can keep, and is called the *carrying capacity* of the environment.

If the population is larger (resp. smaller) than the carrying capacity then the population decreases (resp. increases). Once the population reaches the carrying capacity, then it remains unchanged. Consequently, the single species population model implies that the population eventually converges to and stabilizes at the carrying capacity. Notice that the carrying capacity depends on the amount of food supplied by the environment.

3.2 Algorithm for Replica Density Control

In this subsection, we present an algorithm for the replica density control problem. The algorithm is in-

spired by the single species population model: replicas regarded as individuals of a single species, and a network is regarded as an environment. That is, replicas need to consume food to survive and the food is supplied by agents that created by nodes of the network.

In the algorithm, we introduce time interval of some constant length denoted by *CYCLE*. Behavior of each node can be divided into series of the time interval and each replica a node has is decided individually its next state every the time interval. It should be noticed that the start time of the intervals at different nodes need not be synchronized and that next states of different replicas can be decided at different times.

Figure 1 shows the detailed behavior of nodes and agents in the replica density control algorithm.

The behavior of nodes and agents is simple: each node creates a new agent every *CYCLE* time units (i.e., at the beginning of each time interval). Each agent has a specific amount of food on the initial state and traverses the network with the food. Each agent makes a *random walk* independently: an agent migrates from one node to one of its neighboring nodes with equal probability. When an agent visits node v, node v feeds replicas on node v with food the agent has. The replica can exist during the next time interval if it can be fed a specific amount of food, denoted by RF, during the current time interval. The replica is deleted if it cannot be fed food of amount RF during the time interval.

In addition, each node makes a new replica of replica i if the replica i is fed surplus food of amount RF. This idea derives from the fact that the positive per capita growth $g(t)$ in the single species population model is proportional to the amount of surplus food. This scheme is realized in the following way: each agent stores the surplus food into variable *surplus_food* and continues a random walk after *CYCLE* time units from its creation time. When an agent that has surplus food visits node v, node v feeds replicas on node v with the surplus food the agent has. If the total amount of surplus food the replica on node v is fed is RF, node v makes one new replica of the replica by consuming the surplus food. If the agent has no food and no surplus food, it kills itself (i.e., removes itself from the network).

Now, we consider the amount of food F that each agent should supply. Since each node creates one agent every *CYCLE* time units, the amount $F \cdot n(t)$ of food are supplied on the whole network. The goal of the replica density control problem is to adjust the number $r(t)$ of replicas to $\delta \cdot n(t)$. Remind that the single species population model shows that the number of individuals converges and stabilizes at the carrying capacity $f_a(t)/f$. Thus, the algorithm tries to adjust $r(t)$ to $\delta \cdot n(t)$ by adjusting the carrying capacity to $\delta \cdot n(t)$. Since $f_a(t)$ corresponds to the total amount of supplied food on the whole network $n(t) \cdot F$ and f corresponds to the amount of food RF, the following equation should be satisfied:

$$\frac{f_a(t)}{f} = \frac{n(t) \cdot F}{RF} = \delta \cdot n(t).$$

From this equation, each agent should supply food of amount $F = \delta \cdot RF$.

4 Simulation results

In this section, we present simulation results to show that the proposed algorithm can adjust the replica density.

In the simulation, we assume that each agent repeatedly executes the following actions: each agent stays at a node for one time unit, and then migrates to one of its neighboring nodes by a random walk. We also assume that the migration delay between any pair of neighboring nodes is two time units. The following values are initialized randomly:

- the initial locations of agents
- the initial values of the local clocks(i.e., $time_v$, $time_p$)
- the creation time of replicas on each node v (i.e., $create\_time_i (< time_v)$)
- the initial amounts of food that agents have (i.e., $food_p$)
- the initial amounts of food that replicas have fed in the current time interval (i.e., eat_food_i).

The initial amounts of surplus food that agents have (i.e., $surplus\_food_p$) and the initial amounts of surplus food that replicas have fed (i.e., $eat\_surplus\_food_i$)are set to 0.

In the simulation, a new replica created by a node is allocated on the node selected randomly with probability proportional to their degrees. In real systems, replica allocation is very important to get good performance. In this paper, however, we focus on control of replica density rather than how to allocate replicas effectively on the network. The above allocation can be realized as follows: an agent picks up a new replica on a node, and drops the replica on the visited node after it traverses the network by a random walk during random time units.

We present the simulation results for *random networks* and *scale-free networks*. Scale-free networks are a specific kind of networks such that some nodes have a tremendous number of connections to other

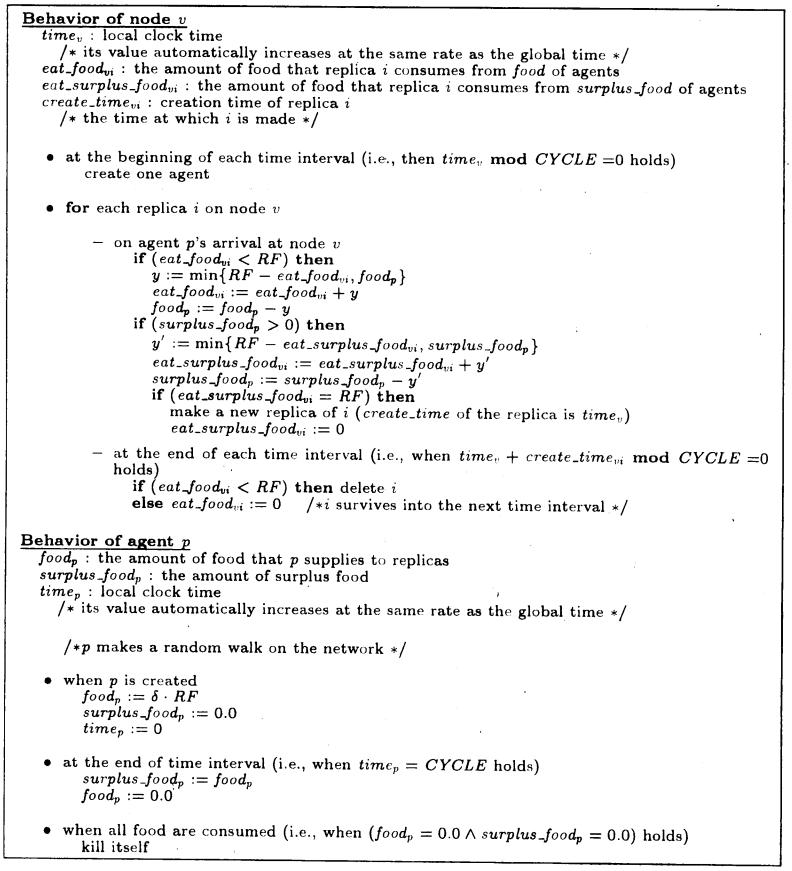


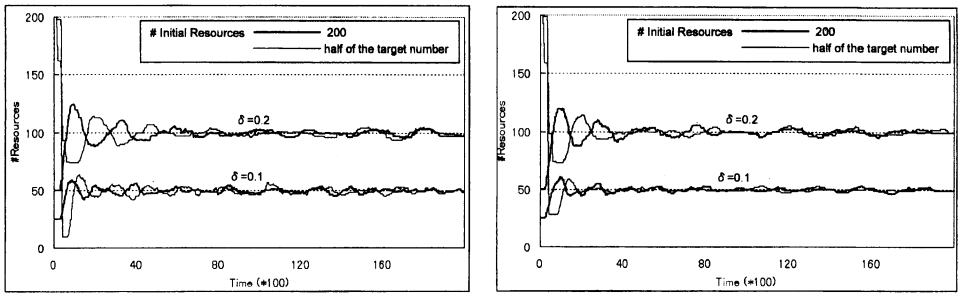
Figure 1: Behavior of node v and agent p

nodes, whereas most nodes have just a handful. The degree distribution follows a power law of k: the number of nodes with degree k is proportional to k^{-r}, where r is a positive constant. A scale-free network is said to be a realistic model of actual network structures [3, 5].

Simulation results for static networks. Figure 2 shows experimental results for "static" random networks and "static" scale-free networks where nodes and links of the networks remain unchanged. In the simulation, the number $n(t)$ of nodes is fixed at 500 during the simulation. Random graphs with n nodes are generated as follows: each pair of nodes is connected with probability of $5.0/(n-1)$. Scale-free networks are generated using the incremental method proposed by Balabasi and Albert [3]. More precisely, starting with 3 nodes, we add new nodes one by one. When a new node is added, three links are also added to connect the node to three other nodes, which are randomly selected with probability proportional to their degrees.

Figure 2 shows transition of the number $r(t)$ of replicas with time t. It shows the simulation results for four combinations of two values of δ (0.2 and 0.1), and two initial number $r(0)$ of replicas (200 and the half of the target number). The length $CYCLE$ of the time interval is set to 200, and the initial number of agents is set to 100. These simulation results show that the number of replicas quickly converges to the equilibrium point, and has small perturbation after the convergence.

Simulation results for dynamic networks. Fig-



a. random networks ($n(t) = 500$)

b. scale-free networks ($n(t) = 500$)

Figure 2: Simulation results on static networks

ure 3 shows the experimental results for "dynamic" random networks and "dynamic" scale-free networks where nodes and links of networks vary with time. When a new node joins in the network, the new node is connected to other nodes with probability $5.0/n(t)$ for each other node on random networks, and the new node is connected to three other nodes randomly selected with probability proportional to their degrees on scale-free networks. When a node v leaves from the network, the links connecting to v are also removed from the network, and replicas the node v has and agents on node v or these links are also removed from the network. To show the adaptiveness of the proposed algorithm, Figure 3 and Figure ?? also show the difference ratio of the number of replicas: the ratio is defined by $|\delta \cdot n(t) - r(t)|/(\delta \cdot n(t))$ and represents the ratio of difference between the adjusted and the target numbers of replicas to the target number. (In static networks, the average of the difference ratio is about 0.02.)

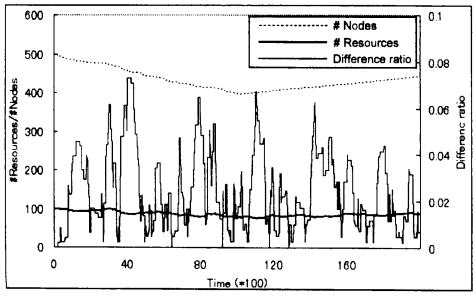
Figure 3 shows simulation results for dynamic networks with continuous and gradual changes: some nodes join in the network and some nodes leave from the network constantly. In this simulation, the initial network size $n(0)$ is 500, and the following dynamical changes occur every 200 time units. In the first half (from time 0 to time 10,000) of the simulation, a new nodes join in the network with probability 0.05 and each node leaves from the network with probability 0.005. In the second half (from time 10,000 to time 20,000), a new nodes join in the network with probability 1.0 and each node leaves from the network with probability 0.001.

In the simulation results of Figure 3, the length *CYCLE* is set to 200, the value of δ is set to 0.2, the initial number of agents is set to 100 and the initial number $r(0)$ of replicas is set to 100. Since the difference ratio is kept to be less than 0.08 and does not widely diverge from 0, the simulation results show that the number of replica is adaptively adjusted in response to changes in the network size.

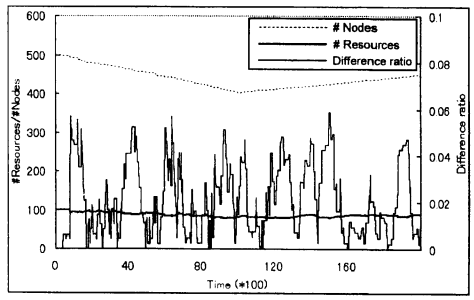
Simulation results on lifetime of replicas. The goal of the replica density control problem is to adjust the number of replicas to a given ratio of the network size. However, from the point of application view, locations of each replica should not be change frequently. In real applications, if there is almost no change of locations of each replica, applications can improve the searching performance. In our algorithm, each replica stays on the same node while the replica exists on the network. Thus, we can say the stability of locations is high by showing that lifetime of replicas is sufficiently long.

Lifetime lt_i of replica i is defined to be the time length from its creation to its elimination, i.e., $lt_i = td_i - tc_i$, where td_i is the time when i is deleted and tc_p is the time when i is created. Table 1 shows the average lifetime of replicas of ten trials. To focus on the lifetime of replicas after convergence of the number of replicas to the target number, the initial number $r(0)$ of replica is set to the equilibrium point. In the simulation results of Table 1, the value of δ is set to 0.2 and the initial number of agents is set to the same number as the initial number of replicas. The simulation results show that lifetime quickly becomes longer when the length of the time interval *CYCLE* becomes longer. Therefore, by setting an appropriate value to *CYCLE*, it is strongly expected that lifetime of each replica becomes sufficiently long.

Simulation results using smaller number of agents. In the algorithm presented in Section 3.2, $n(t)$ agents are created and traverse the network every *CYCLE* time units. Although the size of the agent is so small since the agent has only information



a. random networks
$(n(0) = 500, r(0) = 100, \delta = 0.2)$



b. scale-free networks
$(n(0) = 500, r(0) = 100, \delta = 0.2)$

Figure 3: Simulation results on dynamic networks with gradual changes

		CYCLE		
		100	200	400
n	200	2772	16977	63528
	500	2858	21797	104218
	1000	2915	23347	132340

a. random networks

		CYCLE		
		100	200	400
n	200	3757	18230	62470
	500	4505	27456	97399
	1000	4359	32442	123737

b. scale-free networks

Table 1: Existence time of replicas

of food, the number $n(t)$ of agents may be large for the system. To reduce network traffic, we try to reduce the number of agents by increasing the amount of food one agent has; that is, $1/c \cdot n(t)$ nodes create new agents with food of amount $c \cdot \delta \cdot RF$ every *CYCLE* time units ($c > 1$). Each node picks a number from 0 to c-1 randomly on its initial state and decrements the value every *CYCLE* time units. When the value becomes 0, the node create a new agent that has the amount $c \cdot \delta \cdot RF$ of food and the value is set to $c - 1$. In this regard, however, the length of the time interval *CYCLE* needs to become longer depending on the value of c. The reason is that agents with larger amount of food must visit more nodes to supply food to more replicas. This method reduces by $1/c$ network traffic of agents.

Figure 4 shows the simulation results with small number of agents for "dynamic" random networks and "dynamic" scale-free networks. The value of c is set to 5; about $1/5 \cdot n(t)$ agents are created every *CYCLE* time units. Networks change in the same way as the above simulation of Figure 3. In the simulation, the length *CYCLE* is set to 400, the value of δ is set to 0.2, the initial number of agents is set to 20 and the initial number $r(0)$ of replicas is set to 100.

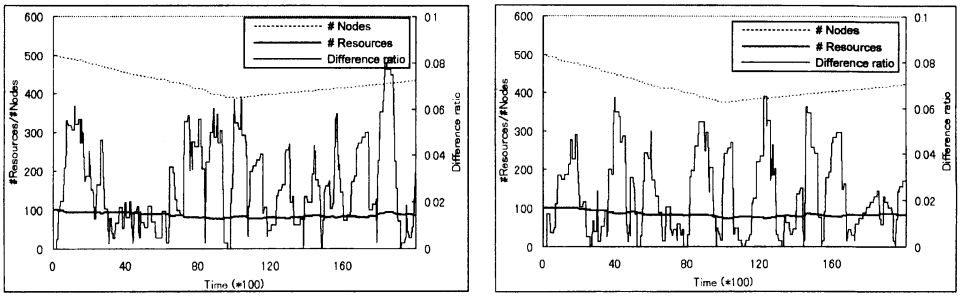
The simulation results in Figure 4 show that the number of replica can be sufficiently adjusted in response to changes in the network size using only small number of agents. Without sacrificing accuracy, the network traffic of agents are reduced to $1/c \cdot n(t)$.

The algorithm we proposed can adjust the replica density even in the case that the value of δ is very small. However, the length of the time interval *CYCLE* needs to become longer when the value of δ is smaller. The reason is that the small value of δ indicates the low density replica and thus, agents must visit more nodes to supply food. We did simulations in small δ and obtained similar results to results in this paper.

Besides the simulations on random networks and scale-free networks presented in this section, we did simulations on several other networks such as complete networks, lollipop networks and star networks, and obtained similar results on these networks.

5 Conclusions

In this paper, we have proposed a distributed algorithm for the replica density control problem that requires to adapt the number of origin resource and its replicas to a given constant fraction of the current number of nodes in a dynamic network. The al-



a. random networks
$(n(0) = 500, r(0) = 100, \delta = 0.2)$

b. scale-free networks
$(n(0) = 500, r(0) = 100, \delta = 0.2)$

Figure 4: Simulation results on dynamic networks using small number of agents

gorithm is inspired by the single species population model, which is well-known in the field of the population ecology. The simulation results show that the proposed algorithm can adequately adjust the number of replicas in dynamic networks. In addition, from the simulation results, the lifetime of each replica becomes sufficiently long by setting an appropriate value to algorithm parameter *CYCLE*.

In this paper, we focus on only the number of replicas. In real systems that provide resource replication, allocation of replicas is also very important. Our future work is to develop the replica allocation algorithm for improving system performance: agents determine allocations of new replicas from network conditions that agents can learn by traversing over the network.

Acknowledgment

This work is supported in part by a Grant-in-Aid for Scientific Research ((B)(2)15300017) of JSPS, and Grant-in-Aid for Scientific Research on Priority Areas(16092215), and "The 21st Century Center of Excellence Program" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] The anthill project. http://www.cs.unibo.it/projects/anthill/.

[2] The bio-networking architecture. http://netresearch.ics.uci.edu/bionet/.

[3] R. Albert and A. L. Barabasi. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47–97, January 2002.

[4] O. Babaoglu, H. Meling, and A. Montresor. Anthill: A framework for the development of agent-based peer-to-peer systems. In *Proceedings of the 22th International Conference on Distributed Computing Systems*, pages 15–22, 2002.

[5] A. L. Barabasi and E. Bonabeau. Scale-free networks. *Scientific American*, 288:50–59, May 2003.

[6] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. In *Proceedings of the Workshop on Design Issues in Anonymity and Unobservability*, pages 46–66, July 2000.

[7] Gnutella.com. http://www.gnutella.com.

[8] R. Haberman. *Mathematical Model : Population Dynamics*. PRENTICE HALL, 1977.

[9] K. Miura, T. Tagawa, and H. Kakugawa. A quorum-based protocol for searching objects in peer-to-peer networks. *IEEE Transactions on Parallel and Distributed Systems*. to appear.

[10] V. A. Pham and A. Karmouch. Mobile software agents : An overview. *IEEE Communications*, 36(7):26–36, July 1998.

[11] A. R. Silva, A. Romao, D. Deugo, and M. Mira. Towards a reference model for surveying mobile agent systems. *Autonomous Agents and Multi-Agent System*, 4(3):187–231, 2001.

[12] J. Suzuki and T. Suda. Design and implementation of a scalable infrastructure for autonomous adaptive agents. In *Proceedings of the 15th IASTED International Conference on Parallel and Distributed Computing and Systems*, pages 594–603, November 2003.

[13] T. Suzuki, T. Izumi, F. Ooshita, and T. Masuzawa. Biologically inspired self-adaptation of mobile agent population. In *Proceedings of 3rd International Workshop on Self-Adaptive and Autonomic Computing Systems*, August 2005. to appear.