

Economics-Inspired Decentralized Control Approach for Adaptive Grid Services and Applications

Lei Gao,^{1,†} Yongsheng Ding,^{1,2,*} Hao Ying^{3,‡}

¹*College of Information Sciences and Technology, Donghua University, Shanghai 201620, People's Republic of China*

²*Engineering Research Center of Digitized Textile & Fashion Technology, Ministry of Education, Donghua University, Shanghai 201620, People's Republic of China*

³*Department of Electrical and Computer Engineering, Wayne State University, Detroit, MI 48202, USA*

Grid technologies facilitate innovative applications among dynamic virtual organizations, while the ability to deploy, manage, and properly remain functioning via traditional approaches has been exceeded by the complexity of the next generation of grid systems. An important method for addressing this challenge may require nature-inspired computing paradigms. This technique will entail construction of a bottom-up multiagent system; however, the appropriate implementation mechanism is under consideration in order for the autonomous and distributed agents to emerge as a controlled grid service or application. A credit card management service in economic interactions is considered in this article for a decentralized control approach. This consideration is based on a preliminarily developed ecological network-based grid middleware that has features desired for the next generation grid systems. The control scheme, design, and implementation of the credit card management service are presented in detail. The simulation results show that (1) agents are accountable for their activities such as behavior invocation, service provision, and resource utilization and (2) generated services or applications adapt well to dynamically changing environments such as agent amounts as well as partial failure of agents. The approach presented herein is beneficial for building autonomous and adaptive grid applications and services. © 2006 Wiley Periodicals, Inc.

1. INTRODUCTION

The concept of “grid” emerged in the early 1990s as a distributed infrastructure to network computers utilized for building virtual supercomputers. To hide the heterogeneous nature and provide users and applications with a seamless grid environment, the issues in designing, building, and deploying grid systems have

*Author to whom all correspondence should be addressed: e-mail: ysding@dhu.edu.cn.

†e-mail: ieloag@mail.dhu.edu.cn.

‡e-mail: hao.ying@wayne.edu.

INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS, VOL. 21, 1269–1288 (2006)
© 2006 Wiley Periodicals, Inc. Published online in Wiley InterScience
(www.interscience.wiley.com). • DOI 10.1002/int.20183



been explored for over a decade. Grid computing is defined as flexible, secure, coordinated resource sharing and problem-solving in dynamic virtual organizations (VOs).¹ To build new grid applications, next generation grid systems are moving toward globally collaborative, service-oriented, and live (real-time) information systems that exhibit a strong sense of automation^{2,3} such as (1) surviving massive failures and attacks, (2) dynamically configuring and reconfiguring systems, (3) containing environmental awareness, (4) seeking behavior optimization to achieve their goals, and (5) requiring detailed knowledge of system components and status. The centralized decision-making and hierarchical organization fails to perform in such environments; therefore, the paradigm needs to shift toward a decentralized philosophy.

Developing an efficient grid service (including application-level service) is becoming more difficult due to the increasing complexity of the grid systems. A critical challenge is faced that allows grid services to autonomously adapt to the changing grid conditions and release service developers from managing and tuning their services. Thus, the importance of designing a decentralized control approach for enabling autonomous and adaptive services has become obvious.

Several technologies that build the next generation grid, such as the Semantic Grid,⁴ the Open Grid Services Architecture (OGSA),¹ and Peer-to-Peer (P2P) computing technologies,⁵ all share similar control approaches either explicitly or implicitly. The Semantic Grid uses semantics and knowledge discovery technologies to allow grid entities to interoperate, dynamically discover and use resources, extract knowledge, and solve complex problems. In the OGSA, grid services are dynamic, transient, and globally distributed without centralized control or a globally agreed trust relationship. It aims at absorbing web service features in order for grid services to be controlled, fault resilient, and secure. From the viewpoint of grids, the important features of P2P are (1) the idea of nonhierarchical decentralized systems, (2) autonomic management, (3) dynamic resource discovery, and (4) fault tolerance. However, the design of a reasonable and effective decentralized control approach is under preliminary investigation and a long way from completion.

Previous work in other fields such as web service composition⁶⁻¹⁰ and multi-agent systems¹¹⁻²¹ have also addressed the problem of enabling distributed entities to create more useful services and applications.

Currently, a growing trend in software architecture is building web services that are distributed and contain platform-independent software components from which applications are to be assembled in an automated way. For this purpose, several control methods have been proposed, such as the workflow technique and AI planning. The former technique relies on flexible workflow and automatic process adaptation to composite services. For example, Eflow^{7,8} is a composite service controlled by a graph that defines the execution information. In the latter approach, a plan that enables dynamic service composition is generated automatically by a logical theorem prover or AI planners without knowledge of predefined workflow. A logic programming language⁹ is used to connect a set of atomic services for forming composite services. Composability rules¹⁰ are designed to generate composition plans. Although these control approaches promote the flexibility and dynamic features of web services, there is no guarantee for application

in large-scale decentralized environments. In addition, these approaches lack self-organization capability and require extensive manual operations.

Multiagent systems focus on the development of concepts, methodologies, and algorithms for autonomous problem solving that is flexible in ambiguous and dynamic environments. Several efforts in this area relate the solution of establishing a decentralized control strategy. Nature and economics have provided direct inspirations of building strategies known as the market-based control¹¹⁻¹⁵ and the nature-inspired approach.¹⁶⁻²¹

A vast number of economic-inspired mechanisms such as bilateral negotiations, auctions, contract net, and bid-based resource allocation have been proposed for solving engineering problems. Among them, the market-based control approaches have been reported to be the most successful. In the market place, participants trade their goods and services with the intention to maximize their own utilities. This results in highly efficient resource allocations in dynamic and uncertain environments. Examples of market-based control can be found in distributed resource allocation, factory scheduling, manufacturing systems, energy distribution, automatic price setting system, and pollution management.¹¹⁻¹⁵ These mechanisms influence the design of distributed and decentralized systems, especially when scarce resources are demanded by populations of consumers. However, these mechanisms focus on solving resource allocation-type problems and do not include the service creation aspect.

Nature-inspired approaches incline to construct a *bottom-up*¹⁶⁻¹⁸ and *natural ecosystems*-inspired multiagent system. This methodology is similar to the approach presented in this article. In these systems, the agents may be simple, but their collective behaviors and overall functionality arising from their interactions exceed the capacity of any individual agent. Jack-in-the-net¹⁹ implements a self-tuning mechanism of service composition. It is based on evaluating the strength of the relationship among agents that reflect user preferences and service patterns. AntHocNet²⁰ is an algorithm for routing in mobile ad hoc networks. The algorithm is based on the nature-inspired ant colony optimization framework. In the algorithm, paths are learned using antlike agents that communicate in a stigmergic way. To interact with agents representing users with similar interests, DIET²¹ uses a genetic algorithm-based approach to develop agent preferences for choosing environments. Although these approaches are effective in special fields of network applications, the research in this domain is still at a premature stage. The approach presented in this article may be included into this domain.

In this article, a new control mechanism is considered that addresses the challenge of the adaptive grid services and applications. Observations reveal the similarity between control mechanisms based on decentralized autonomic decision making in economic interactions and next generation grid environments. The similarity inspires new creation technologies of grid services and applications by associating them with key concepts and principles in economic interactions.

Based on previous compositions,²²⁻²⁴ a *credit card* mechanism is introduced for designing the decentralized control approach. In preceding work, the grid was viewed as a number of interacting agents. Furthermore, it applied key mechanisms of natural ecosystems to preliminarily develop a novel ecological

network-based grid middleware system named Ecological Network-based Grid Middleware (ENGM). A credit card is a payment card issued to a person for repeatedly purchasing goods and services and obtaining cash on credit. It includes two important economic concepts of payment and credit. A *credit card management service* is proposed, which is expected to enable a number of relatively simple agents in ENGM platform to be responsible and explain their activities for achieving their common or group interests. Thus, desired services and applications can emerge from the local interactions among the agents and with the environment.

The rest of this article is organized as follows. Section 2 designs the roles of credit card management in the ENGM platform, based on the decentralized control scheme. Section 3 presents the detailed design and implementation of the credit card management service. Furthermore, to illustrate the work mechanism of the credit card management service explicitly, a workflow demonstration of the service is also specified in this section. In Section 4, two simulation experiments are conducted with the credit card management service to empirically verify the desired services and applications emerging by using the proposed control approach. Section 5 concludes the research efforts.

2. CREDIT CARD-BASED DECENTRALIZED CONTROL SCHEME IN THE ENGM PLATFORM

2.1. The Architecture of the ENGM

The grid's three recognized layers are (1) computation/data, (2) information (in which data produce information), and (3) knowledge (in which knowledge can be used to make decision).³ Considering these layers, the ENGM architecture²² is proposed as shown in Figure 1. In the ENGM, each agent follows a simple set of behavior rules (e.g., migration, reproduction, mutation, and death) in order to implement a functional component related to its grid service or application. Furthermore, a grid service or application can emerge from a collection of agents such as the creatures that live in a large ecosystem.

From the implementation point of view, the architecture of the ENGM system is also regarded as a three-layered model: (1) Heterogeneous and Distributed Resources Layer, (2) ENGM Layer, and (3) Grid Applications for the VOs Layer. The Heterogeneous and Distributed Resources Layer consists of different types of resources that are available from the multiple VOs, such as computing power and storage capabilities. ENGM Layer provides the services needed to support a common set of applications in a grid environment. The ENGM platform can operate on a heterogeneous distributed system that is established in a network node. For special applications such as E-business, ENGM is used for the VOs Layer of grid applications.

The ENGM consists of functional modules, core services, a grid agent survivable environment, an emergent grid common service, and grid pluggable developing kits. In ENGM functional module layer, low-level functions that relate

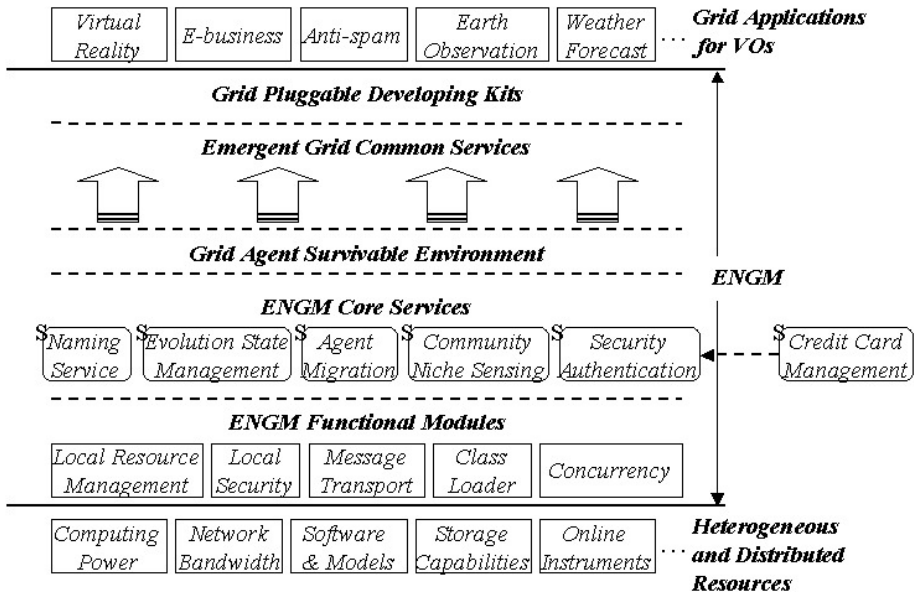


Figure 1. The ENGM architecture motivated by three conceptual layers.

management of networks and systems are dealt with, such as the message transport module. The ENGM core service layer provides a set of general-purpose runtime services that are frequently used by the agents. They include naming service, community niche sensing service, agent migration service, evolution state management service, and security authentication service. These services free agents from low-level operations and allow lightweight agents to be separated from the routine work. The grid agent survivable environment is a runtime environment for deploying and executing agents that are characterized by high demand for computing, storage, and network bandwidth requirements. Emergent grid common services are essential for grid computing and are responsible for resource location and allocation, authentication, information service, and task assignment. These common services emerge from those agents and their invoking low-layered services. Grid pluggable developing kits are sublayers that provide developing environments. They contain low-level function developing, agent creation, remote visualization, and collaborative applications and are pluggable in terms of special grid applications.

The middleware achieves built-in mechanisms to support the features desired for the next generation grid systems such as self-organization, survivability, and self-evolution. It is assumed that the solution outlined in this article can provide the next generation grid systems with the ability to cope with the complexity including survivability, scalability, robustness, and inherently adaptability at the system level.

In this article, the credit card management service inspired from economic interactions is added to the ENGM core services layer that is represented in Figure 1.

2.2. Roles of the Credit Card Management

The components in an ecosystem contact each other to form a functional architecture through energy flow, material flow, and information flow. The maintenance and evolution of the “inside in order” status in the ecosystem must rely on a cycled flow. Assume that money is units that form a cycled flow. If a credit card is set to each component, this enables it to be responsible for its performed behaviors. Similarly, we can also regard credit cards to analyze the ENGM platform system.

In the ENGM, an agent must save and spend money on the credit card. Furthermore, the agent can use the credit card to carry out the roles as shown in Figure 2. These roles include (1) expending money for using resources such as occupying the CPU, memory, and bandwidth. (2) To gain money from the users (or the other agents) an agent needs to perform a service. When a service is not provided, nothing is stored on the credit card although money is expended for resource utilization; therefore, if a service is not performed, money will decrease on the credit card. (3) When an agent provides a service with high quality, additional money is received from a service requester. Furthermore, when an agent provides a service with low quality, money decreases on the credit card. (4) It is possible for an agent in ENGM to evolve, although internal states are required (to be described in Section 2.3). The transitions of agent evolution states are driven by the amount of money on the agent’s credit card. (5) If the money expended is greater than the amount of money earned by providing a service, there will be a money deficiency and a restricted usage of resources. As a result, the agent dies due to inefficiency. (6) Trust is another important concept that is

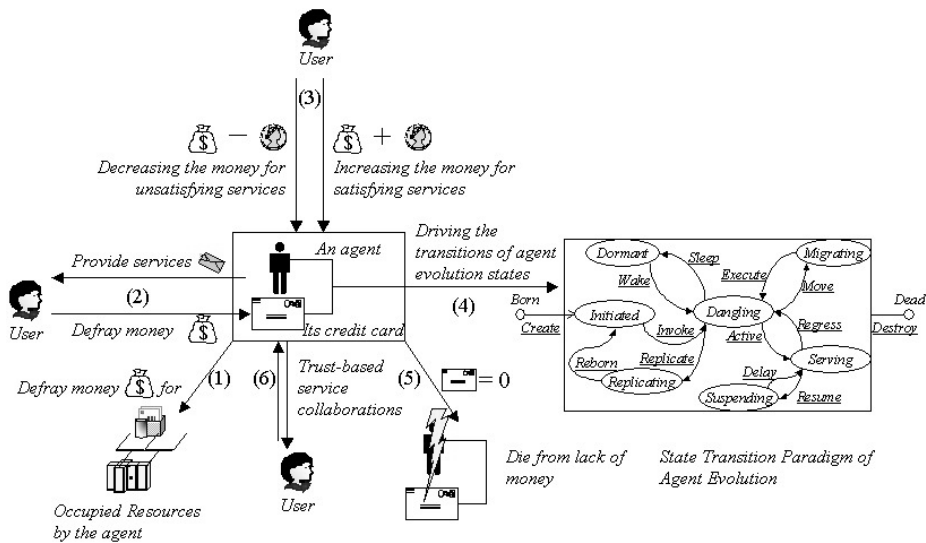


Figure 2. The roles of the credit card management.

included with the usage of a credit card and is included within all agent interactions in the ENGM architecture. For example, if a user (or an agent) wants to seek a service partner, he will demand only reliable acquaintances. Furthermore, in the collaboration, trust between a user (or agent) and an acquaintance is based on positive interactions.

Agent functions and interaction strategies have to be customized for each type of service or application for specified configurations. However, the intent is to find a methodology that forms desirable services or applications that allow each configuration to form various services or applications of the same type. This methodology is considered to contain a credit card management mechanism. It is expected that, by proposing an effective credit card management mechanism, some intractable control issues such as decentralized resource discovery and allocation can be transformed to find a reasonable economic flow strategy in the ENGM architecture.

2.3. Decentralized Control Scheme Based on the Credit Card Management

The decentralized control scheme is relative to the design of agents. Agents in the ENGM can be represented by the formula $An\ agent = Attributes + Evolution\ State + Function\ Unit$. Moreover, an agent consists of three main parts: attributes, evolution states, and a function unit.

The attributes consisted in the metadata of an abstract agent in ENGM are shown in Table I. *agentID* is a globally unique identifier of an agent, for example, GUID (short for Globally Unique Identifier, unique 128-bit number that is produced by the Windows OS or by some Windows applications). *agentAddress* is the location of the agent and the *serviceType* is the variety of service provided by the agent. *serviceDescription* includes the explanation of service information including service price. Finally, *relationshipDescription* relates to the information stored by the agent regarding the agent’s acquaintances (other agents known by the agent).

It is known that agent evolution states maintain the status of an agent. Furthermore, to maintain the activity of the agents within grid environments, mechanisms such as credit card driven and message passing are used to control the transition states of agents. As shown in Figure 2, after an agent is born and before it dies naturally, it will be in one of the following states: *Initiated*, *Dangling*,

Table I. Attributes of an abstract agent in the ENGM platform.

Status	Attribute	Description
Public	<i>agentID</i>	A global unique identifier
	<i>agentAddress</i>	Location in a grid
	<i>serviceType</i>	Provided service type
	<i>serviceDescription</i>	Service description
	<i>trustValue</i>	Indication of reliability to acquaintance
Private	<i>relationshipDescription</i>	Information about acquaintances

Dormant, Serving, Suspending, Replicating, and Migrating. Transitions between different states can be implemented by the transferring operations. As demonstrated in Figure 2, these operations are *Create, Sleep, Wake, Invoke, Reborn, Replicate, Active, Execute, Regress, Move, Resume, Delay, and Destroy.* For example, *Invoke* is used to transform an agent from the *Initiated* state to the *dangling* state.

Function unit implements the service or application function that is provided by the agent, and the implementations depend on requirements of different services or applications. In addition it also contains data that is relevant to the function.

Based on the design of agents, the proposed control scheme is clarified. With reference to the descriptions in Section 2.2, the credit card management contains various roles. The control scheme should specify the implementation manners of these roles in an agent. Each agent has a predefined pricing scenario for its functional unit. In addition, each resource such as the CPU, memory, or bandwidth has individual pricing scenarios. Each service requester (a user or agent) has payment scenarios including rewards and punishments according to the requirements. Each agent state transition may be implemented by one or more simple algorithms. Each algorithm consists of a set of parameters, weights, and a threshold. Due to space limitations, an example of the *Move* operation is given in the transition from *Dangling* state to *Migrating* state as follows.

The ENGM platform uses the migration mechanism of mobile agent systems²⁵ to implement the code mobility. The parameters in the *Move* operation may include *moveBenefit, moveCost, moveRisk, and moveAsynchronous.* A set of weights ($w_1, w_2, w_3,$ and $w_4,$ where $\sum_{i=1}^4 w_i = 1$) indicates the degree of importance of each parameter. The *moveThreshold* is a threshold that determines whether or not the agent migrates.

The agent will perform the *Move* operation if inequality 1 exists:

$$\begin{aligned} & (\textit{moveBenefit}, \textit{moveCost}, \textit{moveRisk}, \textit{moveAsynchronous}) \cdot (w_1, w_2, w_3, w_4)^T \\ & \geq \textit{moveThreshold} \end{aligned} \quad (1)$$

moveBenefit in inequality 1 could be a positive value, and it mainly implies proximity to the service requester, lower prices of resources on the destination node, and lower delays of the network transmission. *moveCost* may be a negative value, and it is defined as the amount of money required for using the network resources for migration. It is also defined as the higher prices for resources on the destination node. *moveRisk* may include some unknown factors or failures in the migration process. An example of this would include similar agents on the destination repelling the migrating agent. In addition, the service delays, which can have a negative value, that are generated from the migration process will be penalized by diminishing the amount of money on the agent's credit card. *moveAsynchronous* is a parameter that assumes the asynchronous migration model and may possibly be a negative value. In the asynchronous model, the migration request is not immediately addressed; however, it is initially inserted into the processing queue. Using this model, agents can operate correctly even if the users are in the low-quality connection or disconnection status most of the

time. It then becomes evident that the agent must compensate for sensing the environments.

Note that the list of parameters provided above is not an exhaustive list of migration parameters for the *Move* operation. Most importantly, each operation of evolution state transitions implements a corresponding algorithm similar to the ENGM platform.

In addition, the requester evaluates the quality of service that was provided by the agent and then gives a credit value to the service provider. This approach will help the service requester determine which agents to trust. Following this, a possible approach is presented that updates the *trustValue* of an agent.

The service requester assigns a credit ζ ($-\frac{1}{2} \leq \zeta \leq 1$) that represents a reward or penalty to the service provider. The trustValue V of the service provider can then be updated according to the following formula:

$$V^+ = \begin{cases} V^-(1 - \zeta^2) + \zeta^2, & \zeta \geq 0 \\ V^- \left(2 - \frac{1}{1 + \zeta} \right), & \zeta < 0 \end{cases} \quad (2)$$

where V^- is a number between $[0, 1]$ that represents the trustValue before the update. Correspondingly, V^+ is the trustValue after updating, and the range of ζ is set to $-\frac{1}{2} \leq \zeta \leq 1$ for ensuring that the value of V^+ can change in $[V^-, 1]$ (when $\zeta \geq 0$) and $[0, V^-)$ (when $\zeta < 0$). More importantly, formula 2 is selected for evaluating the slow rise or quick fall of the agent.

3. DESIGN AND IMPLEMENTATION OF THE CREDIT CARD MANAGEMENT SERVICE

The credit card management service is mainly responsible for the income and expenditure of each agent in the ENGM platform. Agents must store, receive, and expend money through the credit card management service. In this section, detailed design and implementation of the credit card management service in the ENGM platform will be discussed.

3.1. Components of the Credit Card Management Service

Three components of the credit card management service as shown in Figure 3 are established: information center, bargaining bazaar, and market agency. The names and functionalities of these components are also motivated from the economic interactions in the real world. In addition, the Agent Communication Language (ACL) from the Foundation for Intelligent Physical Agent (FIPA)²⁶ is reused. The abstract communication language protocol designed a high-level language entitled Ecological Network Communication Language (ENCL)²³ for the purpose of implementing communication among all agents.

(1) The information center (an information board in a securities market is a kind of information center) in each platform is responsible for the registration,

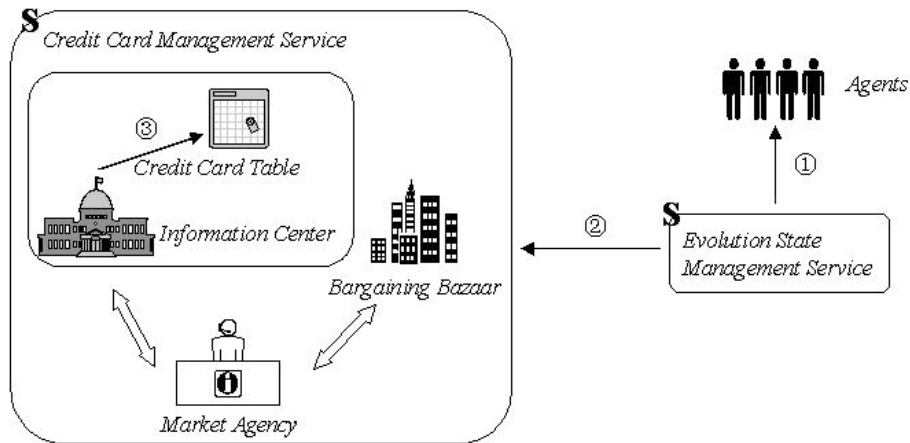


Figure 3. The credit card registration in the information center.

maintaining, and management of a local *credit card table*. The credit card table stores the information of the agents running on the platform. The information includes agent ID, thread name of the agent (each agent runs on an independent thread and its thread name remains consistent for detecting resource utilization), money amount in its credit card, *trustValue* of the agent, and evolution state symbol of the agent. The credit card table is established when the platform starts and destroyed when the platform ends. Evolution state management service of the ENGM can add and delete records of the credit card table. If an agent is created, replicated, reproduced, or migrates from another platform, evolution state management service then registers its record to the credit card table. If an agent dies or migrates to another platform, its record is deleted. The credit card table is read-only for agents, and they can query the amount of money but they cannot modify it. The registration process of new agents is demonstrated in Figure 3, where ① displays the process of initiating agents. The evolution state management service allocates an agent ID, an initial money amount, and a thread name to each agent. ② is the process of asking the credit card management service to register the records of the agents to the credit card table. ③ is the process where the information center adds the credit cards of the agents to the credit table.

(2) The bargaining bazaar is used for the service providers and the service requesters to seek suitable service partners, negotiate the price, and defray the money. It also includes the pricing center and the bargaining tables. The pricing center may adopt some auction or barter models to achieve the harmonious allocation of multiple service resources. The bargaining tables are created based on the ENCL messages seized by a market agency. Every bargaining table has the ability to manage and track the information on money defrayment, including message ID, service provider ID, service provider address, and negotiated price. When the service ends, the bargaining tables will be destroyed.

(3) The market agency is created when the platform starts. It seizes the interacting messages from every agent and then delivers these messages to the bargaining bazaar. Meanwhile, it sends the messages to update the money amount in the credit cards of the agents.

3.2. Workflow-Based Implementation of the Credit Card Management Service

To demonstrate the work mechanism of the credit card management service explicitly, an instance of a service process between *A* and *B* (short for Agent *A* and Agent *B*) is provided, whose workflow is implemented using the ENCL,²³ is as shown in Figure 4.

- (1) By visiting *B*'s attribute, *A* can obtain the service type and price of *B*. Then *A* requests *B* for its service and tells *B* how much it could defray. The *request* message contains its message ID.
- (2) *B* agrees to provide the service to *A*. The *agree* message contains the *request* message ID in the *in-reply-to* field.
- (3) The market agency seizes the *agree* message and reports it to the bargaining bazaar. If the seized *agree* message does not describe the price information, the market agency does not interact with the bargaining bazaar but delivers the message to *A* directly.

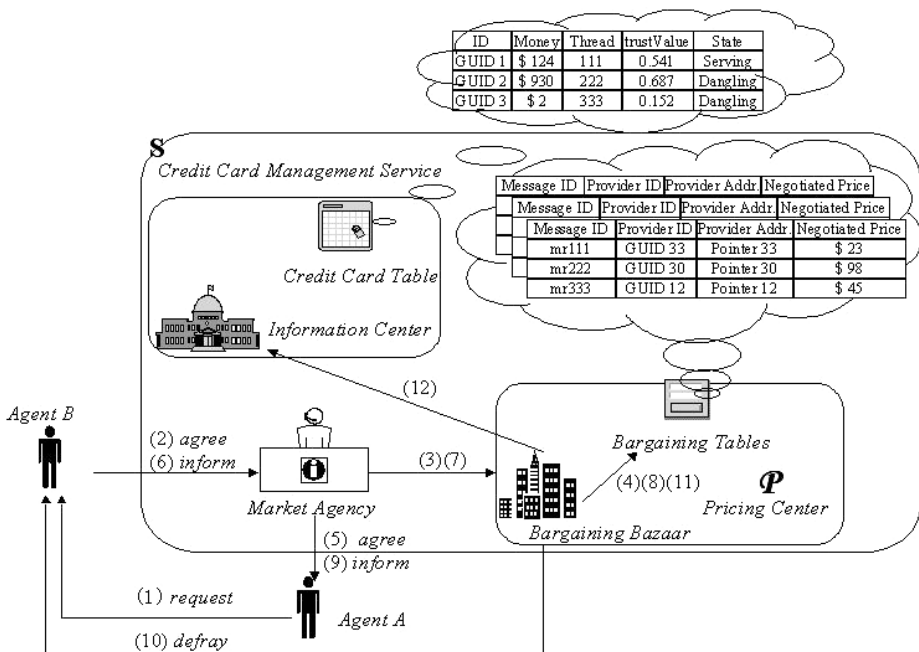


Figure 4. The workflow demonstration of a service process between agent *A* and agent *B*.

- (4) The bargaining bazaar creates a new bargaining table, which includes the *request* message ID in (1), *B*'s ID and address, and the negotiated price that *A* and *B* agreed on in (1) and (2).
- (5) The market agency forwards the *agree* message to *A*.
- (6) Using the *inform* message, *B* provides the service to *A*. The *inform* message also contains the message ID of the *request* message within the *in-reply-to* field.
- (7) The market agency seizes the *inform* message and notifies the bargaining bazaar.
- (8) The bargaining bazaar examines whether the *request* message ID in the *inform* message has been registered in the bargaining table. If the message ID has been registered, it forwards the *inform* message to *A* through the market agency as shown in (9), sends a *defray* message to *B* as shown in (10), and deletes the bargaining table as shown in (11). Then, the bargaining bazaar notifies the information center to decrease *A*'s money for defraying *B* as shown in (12). If the message ID has not been registered, the bargaining bazaar only forwards the message to *A*.
- (9) *B* provides the service to *A*.
- (10) The bargaining bazaar sends a *defray* message to *B*. The message contains the negotiated price that *A* and *B* agreed on in (1) and (2).
- (11) The bargaining bazaar destroys the bargaining table.
- (12) Through the information center, the bargaining bazaar visits the credit card table to subtract the money amount that *A* and *B* agreed on in (1) and (2) from *A*'s credit card. Then the market agency can defray money to *B* (through the *defray* message) and update the credit card.

Note that searching for a service partner in the above process takes place outside the bargaining bazaar. If the search takes place inside the bargaining bazaar, the pricing mechanism may be used. The pricing center ensures that agents compete with others agents and requesters can find suitable partners. Generally speaking, the greater the competition, the more efficient the allocation will be. Conversely, if there is no competition, the pricing mechanism then plays an abnormal role.

4. EMPIRICAL VERIFICATION OF THE DECENTRALIZED CONTROL APPROACH FOR GRID SERVICE AND APPLICATION EMERGENCE

To empirically verify that the desired services and applications can emerge by using the proposed decentralized control approach, two simulation experiments were conducted with the credit card management service in the ENGM platform. One experiment is “the emergence of the grid resource discovery service” in which a decentralized and adaptive discovery service was expected to emerge from the collaborations of the agents. It is a critical activity^{2,27}: Given a description of resources, the autonomous agents can assist each other to return a set of contact addresses of resources that match the description in the decentralized grid environments. The other is “the adaptive grid notification services” that does not rely on the collaboration among agents. Each agent in the simulation accepts the notification requirements of the users and delivers the notifications to the destinations with minimal delay. The agents also avoid overloading the system.

Using the pluggable functions and services supported by the ENGM platform, realistic scenarios of these two simulations were replicated by running multiple agents and multiple platform software on a single computer. In the empirical

verification, the simulation experiments were conducted with Java 2 SDK VMs (version 1.4.1 from Sun Microsystems) and Windows 2000 operation system in PC with an Intel Pentium 4 processor (2.4 GHz) and 512 MB RAM.

4.1. Service Emergence of Resource Discovery

The experiment aims at utilizing the presented control approach to allow the agents to collaborate with each other and find the desired resources. Because the service is collaboration oriented, the trust role is emphasized in the credit card management service. In the beginning, a search instruction from a user is prepared as a request message and sent to one of the agents in the simulation network. If the agent possesses the matched resource descriptions, it responds; otherwise the request is forwarded to another agent until the request hit is returned or the request time is exhausted. It is not feasible for peers to recognize other peers in a wide-area grid; therefore, if Peer *A* is familiar with a subset of the other peers, then it can communicate with Peer *B*, and Peer *B* can deliver the request to one of the peers that it is familiar with.

4.1.1. Simulation Setup

Behavior strategy and price parameters: To simplify the experiment, the agents are only allowed to create, serve, migrate, and die (at a regular interval, the agents are destroyed if the amount of money on their credit cards is lower than a certain threshold). Other biological behaviors such as replication, crossover, and mutation are not simulated. A user pays a fixed amount of money such as \$20 to each agent for each requested resource. After the user has received the requested resource, according to the service quality, the user may increase or decrease an additional amount of \$5 in each agent's credit card. For simplicity, agents do not have to pay for the platform they are on. To migrate to another platform, an agent must pay \$200 to the platform. To obtain more money, an agent tends to forward the request to the agent with a high *trustValue*. Formula 2 is chosen as a method to update the agent's *trustValue*. When the user increases an additional amount of money (indicating the user receives a satisfying service), the credit $\zeta = 0.1$; otherwise, $\zeta = -0.1$. The agents in the request-propagated chain will be given the same amount of credit ζ .

Initial network topology and simulation time: The generation method of network topology proposed in Ref. 28 is assumed. In the simulated network, agents are represented as nodes and the edges connect the pairs of agents that are familiar with one other. Resources can only be requested and passed onto one node at each step. Suppose that at every time there is only one request message sent by an agent and the same message can be sent only once by the request originator during the entire simulation. By definition, a cycle starts from a request message sent by an agent and discontinues when all the relevant messages disappear in the system, and 100 cycles are one generation.

Resource distribution: A request message contains a keyword that described the target resource. A grid resource provided in a peer is also modeled as a keyword.

A set of common resources are prepared (containing 20,000 different keywords) as well as a set of new-type resources (containing 2,000 different keywords completely different from common resources). The experiment was conducted on an unbalanced resource distribution strategy: a few numbers (about 8%) of agents provide most (about 80% of the total) of the keywords, whereas the other 92% of agents share 20% of the keywords. Here, the keywords come from the common resource set. The local keywords held by agents are replaced randomly by 100 new-type keywords per generation.

User requests and evaluations: In each simulation, requests are initiated at a fixed percentage of randomly selected agents and contain keywords for request. The performance of the discovery service is studied via simulation on a biased user request scenario (using the probability 0.8 to ask for a number of special keywords that are about 5% of the total keywords available in the simulation network).

4.1.2. Simulation Results

Considering the randomness in the simulations, the experiments were repeated multiple times. Therefore, the results summarized in this section are the averaged values of the measurements.

Discovery performance was experimented with and is measured in the number of hops that represents the node amount, which is depicted in Figure 5a. In addition, Figure 5a also provides the performance measurement of a random forwarding approach that was used as a comparison with the discovery approach. It is observed that the random forwarding has the lowest efficiency, although it is low cost (no need to store any discovery information in the ENGM platform). Its average number of hops (hops is a measurement of node amount) in 50 generations is a balanced distribution strategy of 55.66 hops, which is about five times greater than the collaboration-based discovery service in the same experimental environment. It is obvious that the proposed approach can improve the discovery performance evidently and adaptively. At the beginning of the simulation, the trust relationships between agents were random and the discovery performance was poor. To obtain the target agent, numerous hops are required. As more simulation cycles elapsed, the agent gradually obtained many similar relationships leading to improved performance in the discovery process.

Considering that agents in a wide-area decentralized environment are likely to be unavailable, an experiment was conducted on the adaptability of the approach in such a condition. When the time is 10, 20, 30, 40, and 50 generations, the following ratios were used: $\xi = 0.0002, 0.002, 0.02, 0.2, \text{ and } 1$, respectively, to set the agents with the smallest amount on their credit cards into the “die” state. From Figure 5a, it is observed that a small fraction of the agents are unavailable and the discovery process controlled by the credit card management service still works, thus improving the agent’s performance.

To evaluate the effect of the relationship network size (i.e., the number of agents) on discovery approach, the simulation was conducted with sizes ranging

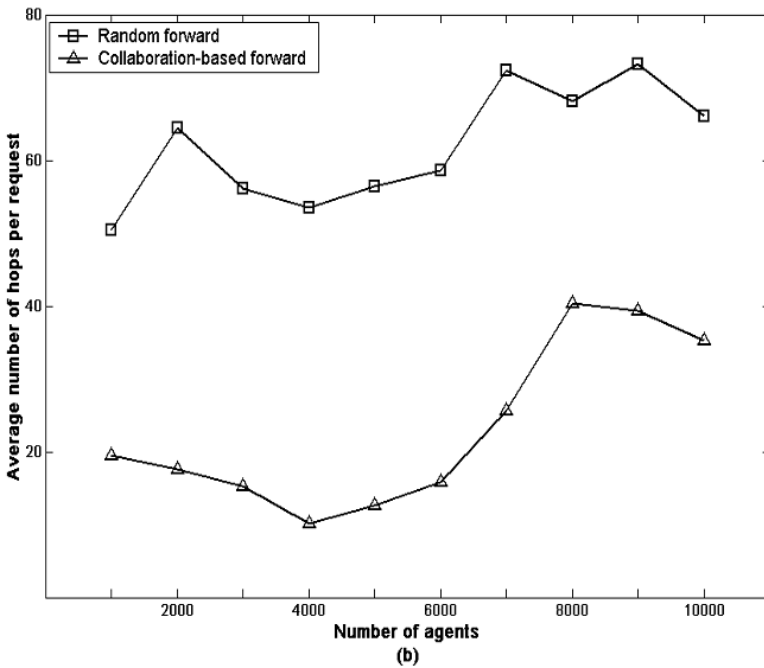
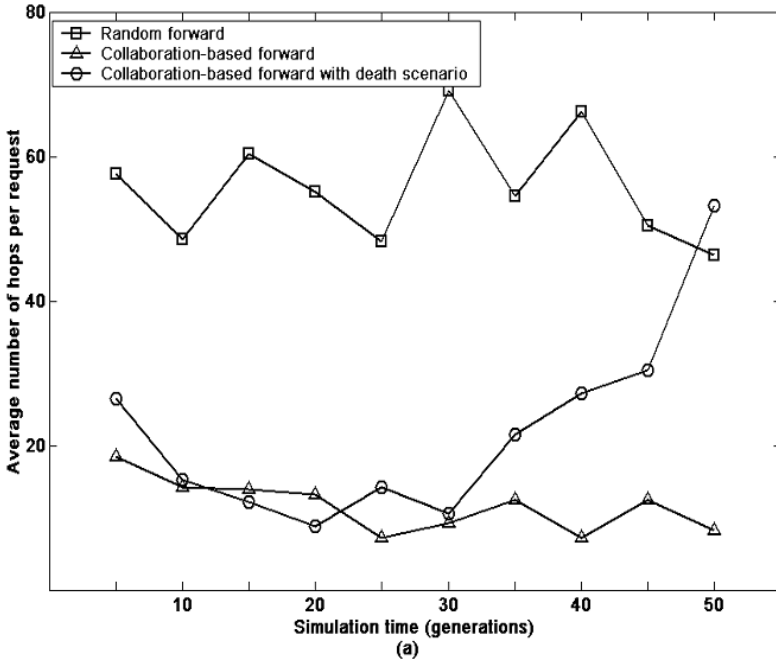


Figure 5. (a) Average number of hops per request as a function of simulation time for different forward strategies. The number of GISA is 5000. (b) Discovery performance with the increase of the number of agents in the simulation setting. The values of TTL in (a) and (b) are both set to 100.

from 1000 to 10,000 and the simulation time was 50 generations. For all network sizes in the experiments, the discovery obtained relatively good performance, as shown in Figure 5b.

Therefore, by exploiting the credit card management service in the ENGM platform, the above discovery process can be properly implemented. The proposed control approach can support a group of agents to emerge as discovery services. In addition, they can adapt to dynamically changing environments such as agent amounts as well as partial failure of the agents.

4.2. Emergence of Adaptive Grid Notification Services

Notification Services play an important role in grid systems.²⁹ They are responsible for asynchronous delivery of messages between publishers and consumers. Although the mechanism for asynchronous is well understood, some issues still remain. For example, the notification services can be used in many aspects of grid service and applications (such as announcement of changes in the content of database, new releases of tools or services, and the termination of workflow execution); however, ensuring that the notification services adapt to the number of user requests and avoid overloading the system still remains a problem. Therefore, the solution proposed in this article is designed to deal with this particular problem.

Every agent in the simulation can provide a simple notification function. The agent can receive the notification contents and requirements from publishers and deliver these contents to consumers. These independent agents distribute geographically in a grid environment to provide the same notification service. Because the application is collaboration independent, this does not imply that there is no interaction among agents. Some factors such as agent behavior strategies or the number of agents contained in a node will affect the decision making of other agents. For example, if there is a large number of agents in a node, other agents will not migrate to the node.

4.2.1. Simulation Setup

Behavior strategy and price parameters: In this experiment, the agents are allowed to create, serve, replicate asexually, migrate, and die. Other biological behaviors such as crossover and mutation are still not simulated. Users pay a fixed amount of money of \$20 to each agent for individual requested notification services. After the user has received the service, the user may increase or decrease an additional amount of \$5 in the credit card of each agent according to the quality of the service. In addition, each agent needs to pay \$150 per second to individual platforms for buying resources (CPU, memory, and bandwidth) to service 10 requests per second. When the demand for resources exceeds the supply, an additional amount of money (\$5 per second) will be added to the resource price. To replicate asexually, an agent must pay \$100 to the platform that it is on. More importantly, the parent agent should give its child a minimum of \$20,000. To

migrate to another platform, an agent must pay an amount of money (\$200) to the platform. Because the application is collaboration independent, the trust mechanism is not adopted in this experiment.

Initial network topology and user requests: A grid topology is situated with 216 nodes (18×12). There are two types of agents in the simulation experiment. One deputizes user requests (like Agent *A* in Section 3.2); the other is a grid notification agent (like Agent *B*). The agents that represent the user requests do not perform biological behaviors. User requests are then placed at random locations in the simulation network. Each user requests one notification service per second. Finally, first priority to serve is given to the grid notification agents that are near the user requests.

4.2.2. *Simulation Results*

At the beginning of the simulation, 60 user agents and 20 grid notification agents are produced. Every grid notification agent has an initial amount of money (\$20,000). The credit card management service periodically queries the community niche sensing service. Furthermore, the price of each resource and resource utilization is checked for each agent on the same platform according to the pre-defined prices of each resource. Finally, the expense from the credit card of each agent is deducted, and if there is no money on the agent's credit card, the credit card management service gives instruction to the evolution state management service to destroy the agent.

In the simulation, a comparison between two experiment settings is conducted: without credit card management service and with credit card management service. Without credit card management service, there is no money flowing among agents; therefore each request must be processed by the agent closest to the user agent. This setting only allows the agents to serve and migrate. The setting with the credit card management service is described as the simulation setup. The request response time and migration frequency of the grid notification agents in these two conditions are evaluated and the results are depicted in Figure 6 and Figure 7. It is observed from Figure 6a,b that the user requests are supplied more quickly in the setting with credit a card management service relative to in the setting without a credit card management service. This is due to the fact that the credit card management service can make the resource allocation structure obtain optimization. Furthermore, the grid notification agents are responsible for their behaviors, and they can migrate and replicate in response to sufficient requests. Therefore, there is a rapid response time with a credit card management service. In Figure 7a,b, comparing the migration frequency in the two experimental settings, there is a decrease in the excessive migration behaviors whereas the response time is shorter within the credit card management service. Considering the costs, some grid notification agents will perform their behaviors deliberately.

In conclusion, the credit card management service can optimize collaboration-independent grid applications. In addition, the decentralized control approach offers a guarantee for adaptability of the ENGM architecture.

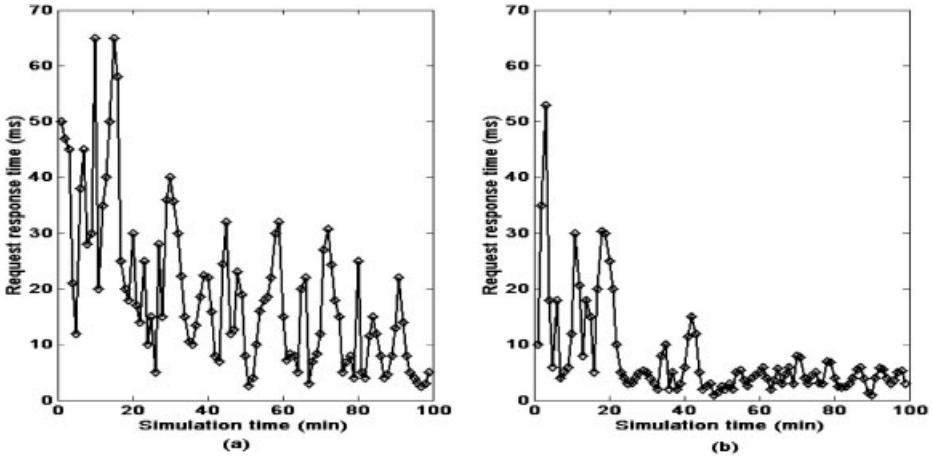


Figure 6. The request response time of the grid notification agents in two experiment settings: (a) without the credit card management service and (b) with the credit card management service.

5. CONCLUSIONS

As grid system capabilities become extended, the complexity has reached a level that exceeds the present ability to deploy, manage, and properly remain functioning via traditional approaches. Important issues requiring investigation are (1) wide-area decentralized grid environments and (2) creating controlled and desirable grid services and applications by determining a mechanism most appropriate for autonomous and distributed entities.

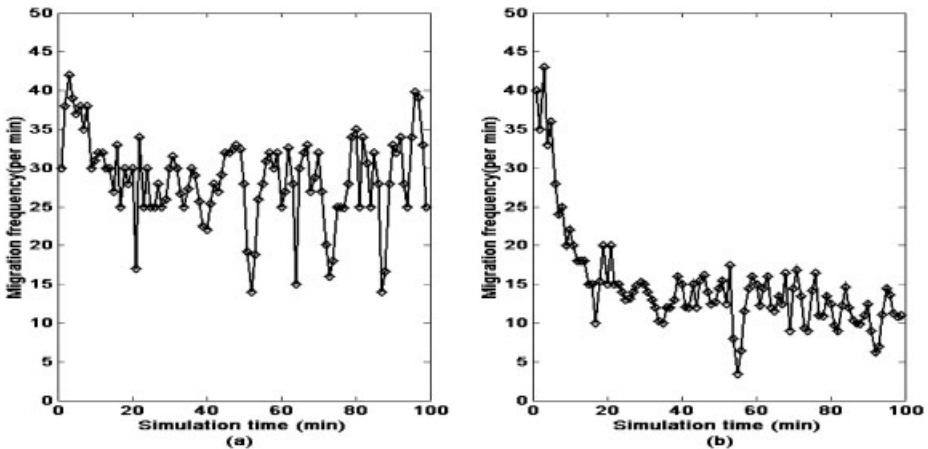


Figure 7. The migration frequency of the grid notification agents in two experiment settings: (a) without the credit card management service and (b) with the credit card management service.

This article has utilized the concepts and principles of economic interactions to design a credit card management service for the ENGM platform. To empirically verify, desired services and applications can emerge using the proposed decentralized control approach. Moreover, two kinds of simulation experiments were conducted: collaboration based and collaboration independent. Although some parameters need to be further evaluated for obtaining more adaptive services and applications, the results of the experiments demonstrate that the proposed credit card management service can hold agents accountable for their activities, such as behavior invocation, service provision, and resource utilization. Thus the grid services or applications emerging from autonomous agents can obtain more adaptability for the changing grid environments.

Acknowledgments

This work was supported in part by the Key Project of the National Nature Science Foundation of China (No. 60534020), the National Nature Science Foundation of China (Nos. 60474037 and 60004006), and Program for New Century Excellent Talents in University (NCET-04-415).

References

1. Foster I, Kesselman C, Nick JM, Tuecke S. Grid services for distributed system integration. *IEEE Comput* 2002;35:37–46.
2. Cannataro M, Talia D. Semantics and knowledge grids: Building the next-generation grid. *IEEE Intell Syst* 2004;19:56–63.
3. De Roure D, Jennings NR, Shadbolt NR. The evolution of the grid. In Berman F, Fox G, Hey AJG, editors. *Grid computing: Making the global infrastructure a reality*, New York: John Wiley and Sons; 2003. pp 65–100.
4. De Roure D, Hender JA. E-science: The grid and the semantic web. *IEEE Intell Syst* 2004;19:65–71.
5. Talia D, Trunfio P. Toward a synergy between P2P and grids. *IEEE Internet Comput* 2003;7:94–96.
6. Milanovic N, Malek M. Current solutions for web service composition. *IEEE Internet Comput* 2004;8:51–59.
7. Casati F, Shan MC. Dynamic and adaptive composition of e-services. *Inform Syst* 2001;6:143–163.
8. Casati F, Shan MC. Event-based interaction management for composite e-services in eflow. *Inform Syst Front* 2002;4:19–31.
9. McIlraith SA, Son TC, Zeng H. Semantic web services. *IEEE Intell Syst* 2001;16:46–53.
10. Medjahed B, Bouguettaya A, Elmagarmid AK. Composing web services on the semantic web. *VLDB J* 2003;12:333–351.
11. Clearwater SH. *Market-based control: A paradigm for distributed resource allocation*. Singapore: World Scientific; 1996.
12. Ygge F, Akkermans JM. Decentralized markets versus central control: A comparative study. *J Artif Intell Res* 1999;11:301–333.
13. Ygge F, Akkermans JM. Resource-oriented multicommodity market algorithms. *Autonom Agent Multi Agent Syst* 2000;3:53–71.
14. Kephart JO, Hanson JE, Greenwald AR. Dynamic pricing by software agents. *Comput Netw* 2000;32:731–752.
15. Kearney P, Smith RE, Bonacino C, Eymann T. Integration of computational models inspired by economics and genetics. *BT Technol J* 2000;18:150–161.

16. Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: From natural to artificial systems. Santa Fe Institute Studies in the Sciences of Complexity. New York: Oxford University Press; 1999.
17. Flake GW. The computational beauty of nature: Computer explorations of fractals, chaos, complex systems, and adaptation. Cambridge, MA: MIT Press; 2000.
18. Holland J. Emergence: From chaos to order. Oxford: Oxford University Press; 2000.
19. Itao T, Tanaka S, Suda T, Aoyama T. A framework for adaptive ubicomp applications based on the jack-in-the-net architecture. *Wireless Netw* 2004;10:287–299.
20. Caro GD, Ducatelle F, Gambardella LM. AntHocNet: An adaptive nature-inspired algorithm for routing in mobile ad hoc networks. *Eur Trans Telecomm* 2005;16:443–455.
21. Marrow P, Hoile C, Wang F, Bonsma E. Evolving preferences among emergent groups of agents. *Lect Notes Comput Sci* 2003;2636:159–173.
22. Gao L, Ding Y-S, Ren L-H. A novel ecological network-based computation platform as grid middleware system. *Int J Intell Syst* 2004;19:859–884.
23. Gao L, Ding Y-S. A flexible communication solution to support grid service emergence. *Lect Notes Comput Sci* 2005;3482:69–78.
24. Gao L, Ding Y-S. Relationship networks as a survivable and adaptive mechanism for grid resource location. *Lect Notes Comput Sci* 2005;3514:517–525.
25. Fuggetta A, Picco GP, Vigna G. Understanding code mobility. *IEEE Trans Softw Eng* 1998;24:342–361.
26. FIPA ACL message structure specification. Foundation for Intelligent Physical Agents. Available at: <http://www.fipa.org/specs/fipa00061/>.
27. Iamnitchi A. Resource discovery in large resource-sharing environments. Ph.D. Thesis. Chicago: University of Chicago; 2003.
28. Barabási A-L, Albert R. Emergence of scaling in random networks. *Science* 1999; 286:509–512.
29. Lawley R, Luck M, Decker K, Payne T, Moreau L. Automated negotiation between publishers and consumers of grid notifications. *Parallel Process Lett* 2003;13:537–548.