A Novel Ecological Network-Based Computation Platform as a Grid Middleware System

Lei Gao,† Yongsheng Ding,* Lihong Ren‡
College of Information Sciences and Technology, Donghua University, Shanghai 200051, P.R. China

Next-generation grid systems where the emphasis shifts to distributed global collaboration, a service-oriented approach, and information layer issues exhibit a strong sense of automation. Requirements for these systems resemble the self-organizing and the self-healing properties of natural ecosystems. Inspired by this resemblance, we introduce some key ecological concepts and mechanisms into the design for the third-generation grid systems. In this article, a novel Ecological Network-based Grid Middleware (ENGM), which is based on Ecological Network Computing Environment (ENCE), is proposed. First, we discuss how to design the ENCE by agent-oriented approaches based on the key concepts and principles of ecosystems. ENCE provides a new computing and problem-solving paradigm by combining natural ecosystem mechanisms with agent technologies. Then, we design the ENGM with built-in mechanisms to support desirable requirements of new grid systems, namely scalability, adaptability, self-organization, simplicity, and survivability. Based on Jeffery’s conceptual model, we also present a corresponding grid-computing prototype that embeds ENGM layers from the implementation point of view. The ENGM will be useful to address the challenges of the third-generation grid systems. Finally, as a demonstration, we built an ENGM platform-based commercial grid service environment and developed a prototype of enterprise supply chain management system. The experimental results demonstrate that the proposed ENGM satisfies the requirements of the next-generation grid and is suitable for new generation grid applications. © 2004 Wiley Periodicals, Inc.

1. INTRODUCTION

Grid technologies have emerged to enable large-scale flexible resource sharing and innovative applications among dynamic virtual organizations.¹ The issues in designing, building, and deploying grid systems have been explored over several years. An increasing number of research groups have been working in this wide-area distributed computing field. These groups have implemented middleware...
systems, libraries, and tools that allow the cooperative use of geographically distributed resources to be unified to act as a single powerful computer.2

The evolution of grid systems is identified to be three generations3: The first generation systems involved solutions for sharing high performance computing resources, such as I-WAY4; the second ones focused on middleware to hide the heterogeneous nature and provided users and applications with a homogeneous and seamless grid environment, such as Globus5 and Legion6; and the next or the third ones shift the emphasis to distributed global collaboration, a service-oriented approach and information layer issues. The third generation systems exhibit a strong sense of automation7: (1) support for survivability from massive failures and attacks, (2) ability to configure and reconfigure system dynamically, (3) awareness of grid system environment, (4) seeking of behavior optimization to achieve its goal, and (5) requirement for detailed knowledge of system components and status.

The requirements of the third grid systems resemble the self-organizing and the self-healing properties of natural ecosystems or biological systems. From the viewpoint of ecology, the grid can be regarded as an enormous growing ecosystem where interaction of its components makes up a huge ecological chain. There is a strong similarity between the complex interaction of organisms in a natural ecosystem and that of components in a networked system.8–10 This prompts us to study the relationships among components in the grid environment by associating it with an ecosystem, especially with some key concepts and principles in it. In this way, we can use the ecological research approaches and introduce some ecological mechanisms to study evolutionary grid systems with those desirable properties.

Grid systems can be regarded as open, giant, distributed, dynamic, and evolutionary heterogeneous information processing systems. Due to their complexity, current popular technologies used to construct large-scale information systems have difficulty addressing such challenges. As such, the complexity theories could provide the methodologies for novel grid systems. On one hand, research on complexity demonstrates that large populations of simple, identical units have the ability to self-organize, form patterns, store information, and reach “collective decision.”11 Even with rather simple individual rules, nonlinear interactions can lead to complex and nonintuitive behavior in large groups.12 On the other hand, the research field of complex systems has become much more sophisticated and diversified.13

For the design of a new generation grid systems, we can identify that at least two approaches will be favored: Artificial life14 and synthetic ecology.15 Artificial life is about the design of artificial creatures that display lifelike properties such as self-reproduction, evolution, and self-organization. It is the crucial complement of ecological and biological research approaches. This approach is more interested in the emergent phenomena than in the optimization of a predefined strategy. Synthetic ecology attempts to apply the functions of natural ecological mechanisms, such as food web laws, into the design of complex systems to make these systems own “automatic” and evolutionary properties.8–10,16 These two approaches largely overlap with each other, but still have quite different goals.

And, from the viewpoint of software implementation, agent-oriented computing technology abstracts complex systems as a collection of interacting, autonomous agents to afford software engineers a number of significant advantages
over contemporary methods. Agent technologies are also viewed as a key framework of designing and constructing intelligent entities to synthesize all branches closely in artificial intelligence. In an agent-based computing paradigm, it has the following features that are competent for desirable requirements: (1) autonomy—agents operate without intervention and have some control over their actions and internal states, (2) social ability—agents interact with other agents using a kind of agent communication languages, (3) goal driven—agents exhibit goal-directed behavior, and (4) reactivity—agents perceive and respond to their environment. These features are particularly well suited to a dynamically changing environment which is one of the important properties for the third-generation grid.

In our previous work, by considering the aspects of complexity theory and agent implementation, we regarded services and applications on the Internet as a number of interacting agents and applied some key features of natural ecosystems to build ecological network computation models as control or selection criteria. Based on these models, we have built a new Ecological Network Computing Environment (ENCE) and its simulation platform, which has the capability of service emergence, evolution, and so forth. The platform can be used to simulate some complex services and applications for the Internet or a distributed network.

In this article, we further study the ENCE and propose a grid computing-oriented prototype. Based on this prototype, a novel Ecological Network-based Grid Middleware (ENGM) is developed to address the challenges of the third-generation grid systems. Section 2 provides an overview of some key concepts and principles in ecosystems to be applied in the ENCE and ENGM. Section 3 discusses the ENCE and its platform. Section 4 describes the building of a prototype of ENCE-based grid computing and discusses the key issues of ENGM. Section 5 provides an application demonstration about enterprise supply chain management using ENGM. Section 6 concludes our research efforts and points out the future work.

2. KEY CONCEPTS AND PRINCIPLES IN AN ECOLOGICAL NETWORK USEFUL FOR GRID COMPUTING

The ecological network is an evolutionary result of natural laws. It is made up of a biological system and an environmental system. A biological system includes producers, consumers, and decomposers. Components in the ecological network contact each other to form a functional architecture through energy flow, material flow, and information flow. Some key concepts and mechanisms in the ecological system can be applied to design adaptive and automatic services and applications in Internet-scale distributed systems like the grid systems.

(1) Emergent properties and behaviors. Each biological individual in ecological network systems performs simple local actions (e.g., migration, replication, and death) and interacts with others. Although a colony of these individuals will exhibit complex emergent behaviors (e.g., adaptability, self-evolution, self-organization, and healing) that the system has, none of its parts have them. Examples include behaviors in ant colonies, the culture created by people of a region,
and the society economy.\textsuperscript{20–22} Emergent computing is generally characterized by these autonomous interactions of relatively simple individuals, forming a system whose whole can be more than its parts. Emergent computing systems are decentralized (or “bottom-up”) and there is usually no central control governing the entire system’s behavior. An emergent algorithm produces global systemwide properties that result from the collective actions of the participating actors. These global properties may be arbitrarily complex, but result from the interactions of large numbers of individual actors, each autonomously performing simple local actions with clearly defined protocols of neighbor interaction. These mechanisms can be applied to realize a secure, adaptive, self-healing, and survivable grid middleware and its services.

(2) Food web rules.\textsuperscript{23} In an ecosystem, the linear succession of the biological system elements (producers, consumers of the first order, consumers of the second order, etc.) is called the food chain. A food chain can be described as a transfer of energy from one organism to another one. The conditions in natural ecosystems are mostly more complex, so that \textit{food web} is a better way to describe the actual state. An ecosystem is an open energy system and a food web is its common model that regards species as the basic analytical unit. A food web describes feeding relationships of one animal or plant to the other members of the community. Feeding relationships can cause invasions, extirpations, and population fluctuations of a species to dramatically affect other species within a variety of natural habitats. It still indicates intuitively that the diversity of the species is of extreme importance to the stability of the ecosystem. The shorter the food chain is and the more simple the structure of the food web is, the worse the system’s ability to provide anti-interference. In a grid computing environment, different components can be regarded as different species in the ecological environment and the relationship among them is a kind of cross feeding one similar to a food web.

(3) Importance of energy. In ecosystems, the maintenance and evolution of the “inside in order” status must rely on the input of external energy. Energy is required for an animal or plant to carry out its life processes. Energy is the driver and the life flow of ecosystems. As we know, food as a part of energy is the material to supply the human body with required energy and necessary nutrition for metabolism. This makes us regard energy as a thread in analyzing the ecological grid systems.

(4) Diversity and evolution of species. A community can be viewed as a collection of numerous environmental variables and functional variables. Each species in a community plays a different role. Predominant species occupy larger space, utilize more resources, and store more energy. Diversity of species is one of the important indexes of community stability. The more richness of species is in a community, the greater the diversity is. Moreover, the same communities in different environments may appear with different structures and adaptive physiological characteristics. Evolution occurs as a result of the natural selection from diversity in biological species. Through many successive generations, beneficial features are retained while detrimental behaviors become dormant or extinct, and the ecosystem specializes and improves itself according to environmental changes. The mechanisms of diversity and evolution of ecosystems can be applied to the optimization of grid service.
(5) Life entity behavior. To fit the survival environment well, a life entity such as a biological individual or a biological population will autonomously perform some behaviors. For example, bird flocks often migrate to warm places for more food. In natural genetics, the parent generation population may produce an evolutionary offspring population by genetic operations such as crossover and mutation. Some behaviors of life entity will be used for designing grid entities’ evolutionary operations.

There are still other ecological characteristics related to the construction of Internet-scale distributed services, which can be referred to. These key concepts and principles will be employed for integrating and coordinating widely distributed resources in the grid environment.

3. ENCE AND ITS PLATFORM ARCHITECTURE

3.1. ENCE

In terms of the key concepts and principles of ecosystems, we discuss how to design the ENCE by agent-oriented approaches. ENCE draws technologies from many parts of agent research including agent autonomy, mobility, multi-agent interaction, and agent abstraction. The ENCE can be used to design, integrate, and manage the complexity of distributed applications by providing a simple natural metaphor for the interaction of the computational components. In ENCE, network services and applications are implemented by a collection of distributed, autonomous mobile agents like the creatures living in a larger ecosystem. We believe that the software agent technique is an important abstraction for building distributed systems. A software agent is an excellent building block for complex open-ended network applications. This methodology will be shown to be a natural way of building distributed computation, and is particularly well-suited to the open wide-area (Internet-scale) environment.

In ENCE, each agent follows simple behavior rules (such as migration, replication, and death) and implements a functional component related to its service. For instance, such an agent or a group of agents can serve as an emergency message broadcaster, an advertising broker, or a survey questionnaire collector. And an application is created out of the interactions of multiple such agents across a network. So in ENCE, agents are more than just distributed, autonomous, and mobile objects with an execution thread; they are the building components of an active distributed system. From this viewpoint, they should have the following properties: (1) They should be able to communicate with other agents; (2) they should be able to form a community to complete a task through interactions with other agents; (3) they can perform different behaviors under a certain control mechanism; (4) they have some evolution states to record the computation status; (5) they can be sent to a system and entrusted to carry out goals; (6) they must have the capability of self-description that can be used to describe and discovery available services.

With the collection of the above distributed, autonomous, and mobile agents, we can build the ENCE with the following characteristics.
1) Service emergence of community. The environment where these agents live is a wide-area (Internet-scale) distributed network. Agents are distributed automatically or semi-automatically via some communication paths. Therefore, agents meet each other and assemble. In ENCE, a collection of agents will form a community as to a certain law, that is, the agents providing similar or complementary services will form a community. A community is able to emerge as a higher-level service through autonomous and self-organizing agent interactions in it. Also these formed communities can be viewed as emerging agents that organize a higher level community to provide complex useful services, as shown in Figure 1.

2) Energy-driven mechanism. The living of all agents depend on energy, which is an important concept inspired by the natural ecosystem. Agents must store and consume energy for their living. They expend energy for their usage of resources. To gain energy from users or other agents, an agent may perform a service. Viewed from this point, energy is similar to money in the economic world. In addition, the changes of agent actions and the transitions of agent evolution states are driven by energy. For instance, the more abundant energy an agent has, the higher demand it needs. If the energy expenditure of an agent is more than its energy earning by providing a service, the agent will lack energy and lose the permission to use resources. Thus, it dies from its wasteful energy performances.

3) Agent evolution states in ENCE. An agent in ENCE can evolve. It can react to an environment, respond to another agent, and communicate with other

![Figure 1. Service emergence of community.](image-url)
agents. The evolution process of an agent involves some internal states. Agent evolution states keep the status of an agent. To maintain the activity of agents, in a distributed computing environment, energy-driven and message passing are used as mechanisms to control agent state transitions. As shown in Figure 2, after an agent is born and before it dies in nature, it will be in one of the following states:

1. *Initiated*: An agent is born with an Initiated state, and then it will shift to a Dangling state.
2. *Dangling*: The agent is ready for receiving messages from other agents, but it does not perform behaviors (i.e., messages are queued, but they are not processed immediately). Each dangling agent consumes memory space, and continuously expends energy for using it.
3. *Dormant*: The dormant state is a “sleeping” state in which an agent is externalized into a file with an object serialization mechanism. A dormant agent does not expend energy because it does not consume any resources.
4. *Serving*: The agent is providing a certain service. It runs on an individual thread and expends energy for the thread and memory space.
5. *Suspending*: The agent is waiting for continuing its service that is delayed due to the changing environment and requirements.
6. *Mutating*: The agent is changed with a new goal and provides a new service.
7. *Migrating*: The thread pool defines the maximum number of threads and avoids creating an unlimited number of threads. If there is no idle thread in a thread pool when an agent tries to become serving, the agent will need to wait until an idle thread is available. If it does not want to wait, it may migrate to a nearby platform whose thread pool has idle threads.

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

(4) Community niche. A community niche refers to a logically defined area where agents in a community can learn their surrounding environment. For instance,
an agent may sense which agents are in the niche, what services they perform, and which resources it can access. This helps an agent to create a new service or join an existing community. The information, such as user information and resource information in the niche, may also be obtained by an agent. Multiple determinants are used to define a community niche, such as the variety of communities and physical proximity among network nodes. Note that different community niches may partly or entirely overlap. Several niches may merge into a unified one where all agents share their information.

(5) Self-description of agents. To interact well to provide better service, it is necessary for an agent to discover (or sense) its desirable service provided by another agent. Agents are deployed as loosely coupled (i.e., independent of other agents), which enables them to discover and invoke each other (including newly created agents) without recompiling or changing any line of code. And an essential adjunct to agent discovery is agent service description. ENCE uses DAML (DARPA Agent Markup Language service) as the basis for designing agent service description. DAML service allows the definition of classes of related services, which makes service reuse more feasible because agents are better able to reason about the relationships between services.

(6) Implementation of agent and community communication. Communities implement the construction of various complex services. Searching, assembling, organizing and coordinating of agents to form a community depend on effective communication and cooperation mechanism. And it is the foundation of realizing the flexible and complicated coordination tactics to use message communication. We reuse “FIPA (Foundation for Intelligent Physical Agents) ACL (Agent Communication Language),” an abstract communication language protocol, to design a high-level language called ENCL (Ecological Network Communication Language). As an extension of ACL, ENCL not only enables flexible communications among agents and communities, but also satisfies our requirements for an open and future-proof environment. Also, we use XML as the primary coding language of ENCL because of its strong capability for data description and metadata aspect. By combining the advantages of the ACL and XML, ENCL is simple, flexible, and can implement the complicated coordination strategies in ENCE. Moreover, through ENCL, we are able to define a generic API to communications, and to support RMI over IIOP (Remote Method Invocation over Internet Inter-Orb Protocol) as a transport mechanism, which is indicated in Figure 3.

(7) Food web-based interactions in ENCE. To offer a better service, agents can share information and cooperate with each other. However, if the resource (e.g., network bandwidth) or service request is insufficient, agents will compete with each other. These phenomena are similar to those in the ecosystem of the real world. The natural world has built a law of balance and the food web embeds the law of creature evolution. We design a general policy based on food web rules to decide the interactions between agents under different conditions. A directed graph that represents the dependency between agents is constructed and some selective strategies are designed to adopt adaptive interactions for agents. By these interactions, agents may autonomously form communities or affiliations. For example,
agents providing complementary services may interact to form a community for cooperation.

3.2. Design of ENCE Platform Architecture

ENCE provides a new computing and problem-solving paradigm by combining natural ecosystem mechanisms with agent technologies. It has facilities to support agent mental state representation, negotiation, communication, and cooperation based on ecological laws.

ENCE is also a software framework fully implemented by Java language. We design an ENCE platform architecture as shown in Figure 4. The layered infrastructure includes Agent Survivable Environment, ENCE core services, ENCE Low-Level Functional Modules, Java Virtual Machine, and a heterogeneous distributed system established in a network node for designing, implementing, and deploying ecological network-based wide-area network services. Obviously, each ENCE platform is hosted in a network node and runs using a java virtual machine which runs on an active operating system.

(1) The Agent Survivable Environment layer provides users with a uniform view of their working environments. Also, it is runtime environment for deploying and executing agents. Each agent lives on Agent Survivable Environment to provide its functional service and perform its behaviors.

(2) The ENCE Core Services layer provides a set of general-purpose runtime services that are frequently used by agents. ENCE Core Services include naming service, community niche sensing service, agent migration service, persistency service, evolution state management service, interaction control service, energy-driven control service, and security authentication service. These services alleviate agents from low-level operations and also allow agents to be lightweight by separating them from routine work.
Naming service\(^{28}\) is capable of tracing agents that move in the global Internet environment and requires basic mechanisms to assign globally unique identifiers (Guids). It derives from care-of mechanisms for locating mobile agents. The care-of for any agent to be traced should be located where it was first created. Agents in ENCE have Guids independent of their locations. But Guids are usually very low-level identifiers. Naming service can associate an entity (an individual agent or a community) with several names that are Guid aliases and suitable for service developers and users. It not only translates a high-level name into the corresponding Guid, but also maintains information about current entity location.

In addition to providing basic naming service, ENCE architecture integrates a community niche sensing mechanism. The community niche sensing is a dynamic discovery service that provides the default solution for information lookup (i.e., service information, resource information) within a community niche. A broadcast protocol registers and deregisters information to the sensing service in the community niche. Besides a proprietary sensing protocol, ENCE has implemented a small-world clustering mechanism\(^{29}\) based on similar keywords as a sensing protocol.

Agent migration service is responsible for sending and receiving an agent to and from another host. It serializes both agent code and execution state into streams suitable for network transfer and storage persistency.

Persistency service lets application developers and system administrators suspend agent execution by storing the agent’s state on a disk. It uses agent migration service to serialize both agent code and data, and saves the information in
persistent storage. Persistency minimizes the consumption of system resources while agents wait to provide a service. In addition, **persistency service** provides the function of fault tolerance by duplicating and storing agent copies before starting critical operations.

As described in Section 3.1, there are seven possible internal evolutionary states for an agent after its birth and before its death. **Evolution state management service** manages internal evolution states of agents. It provides the operations to change the states of agents, such as wake, invoke, reborn, mutate, active, execute, and move agents. In a basic life cycle of an agent, it is not necessary for it to experience every evolutionary state.

Agents communicate with each other using ENCL to enable flexible interaction. We study food web rules, and then propose a general food web-based policy to design different interaction modes among agents. Each agent defines one or more interaction modes that specify a sequence of actions and conversation with other agents. Note that there are two problems: (1) To complete a task, each agent needs to select one or more “appropriate” agents to interact with; and (2) before these agents start a conversation, they need to decide an “appropriate” interaction mode for their conversation. In ENCE, **interaction control service** provides some selective strategies for choosing “appropriate” agents and interaction modes.

As a control and natural selection mechanism, energy maintains the execution of ENCE. Each agent will be responsible for its performed behaviors. **Energy-driven control service** provides a local energy table to store and evaluate energy value of each agent in ENCE platform. Agents must store, receive, and expend energy through **energy-driven control service**. Therefore, it is necessary for this service to adopt a strong mechanism to protect energy from illegal and malicious change.

**Security authentication service** utilizes digital signature to verify that the mobile code has not been altered. In this technique, a local site administrator is required to register public keys from trusted sites to a local database before a local machine receives a code from a remote host. The users of the ENCE platform may accept or deny a migration request from an unknown host. In addition, they are given the capability to preset configuration to accept migration request from trusted hosts.

(3) In the **ENCE Low-Level Functional Modules** layer, **local resource management** and **concurrency** modules manage resources of networks and systems such as I/O, concurrency, and network connection. **Message transport** takes care of marshaling/unmarshaling of messages issued by agents and dispatches incoming messages to appropriate target agents. The **class loader** dynamically loads class definitions of agents into a Java virtual machine when they migrate from one ENCE platform to another. **Local security**, built upon a Java virtual machine environment, provides dynamic authentication and flexible access control.

The ideal model would place an ENCE platform on every device as a network node. **ENCE Low-Level Functional Modules** are just a bridge to maintain access to local resources. However, ENCE platforms require a fair amount of common resources such as CPU power and memory, so they currently run on desktop-sized computers or more advanced computers that proxy for several devices.
3.3. Software Features of ENCE

ENCE is a novel computing and problem-solving environment where an application or a service is created from the interaction of multiple agents and the interaction of agents and their environment (i.e., ENCE core services). Services and applications built using the ENCE platform will share a common set of important characteristics, namely self-organization, survivability, self-evolution, scalability, adaptability, and so on. These characteristics applied in a grid field will be discussed in Section 4.4. Here, we discuss the features of the ENCE platform from the software development viewpoint as follows.

1) Quick development and easy implementation. To developers, constructing an ENCE agent is relatively easy because only simple behaviors and service functional descriptions need to be designed. Technically, such an agent is a combination of a Java object, an execution thread, a remote interface for network communication, and a self-description. These four simple pieces together create an agent, a full-fledged autonomous process in distributed system.

2) Support for building distributed service. ENCE can be distributed on different network node, such as mainframe, workstation, computer cluster, and PDA. It hides all implementation details from users and presents a unified computing environment for users.

3) Dynamic reconfiguration of application. Agents can reconfigure their behaviors based on ecological laws proposed in order to adapt to dynamic network conditions. In addition, subsystems and internal components can be reconfigured to speed up software development, reduce time to market, and save resources.

4) Support pluggable functions and services. We provide a generic and easy-to-use programming API, which can be used by a programmer to develop new tools and applications that ENCE supports. What is more, other common programming languages will be able to use ENCE through a planned remote procedure calling technology such as XML-RPC interface.

5) Open and future-proof environment. ENCE use the speech action-based FIPA ACL to support interoperability among agents and agent-based applications, so it can communicate with any agent system complying with FIPA. In addition, the primary supported encoding of ACL is XML and primary implementation language is Java. They are both open and promising standards. We believe ENCE would hopefully be integrated with future developments.

4. AN ENCE-BASED GRID MIDDLEWARE (ENGM)

It is apparent that service-oriented approaches could provide the flexibility required for the third-generation grid. An important and innovative one among them is the Open Grid Service Architecture (OGSA) framework. From the OGSA viewpoint, the grid can be composed of services. A service is defined as a network-enabled entity that provides some capability, such as computational resources, storage resources, and programs. Core services will implement the interfaces and behaviors described in the Grid Service specification. Basic services will use the Core services to implement some specific services, such as resource management,
data transfer, and resource reservation and monitoring. A range of higher level services will use *Core services* and *Basic services* to provide some information services such as data management, workload management, and diagnostics services. However, OGSA does not provide a new solution or a new technology to address many of challenges of next-generation grid systems. Hence it is valuable to develop other service-oriented models. We believe that ENCE, as such a model, will be well matched for the requirements of the new grid systems. ENCE is a distributed computing environment combining ecological mechanisms and agent technologies. In ENCE, software agents can be seen as producers, consumers, and, indeed, brokers of services. Also many advantageous features of ENCE discussed in Section 3.3 (some ecological features such as survivability, scalability, self-evolution, and self-organization will be discussed in Section 4.4) will be competent for the challenges of the third-grid systems. In the following, we design a novel ENCE-based Grid Middleware (ENGM). Its prototype refers to the three-layered model for the grid proposed by Keith Jeffery.\textsuperscript{33} We also discuss several key issues of ENGM.

### 4.1. Three Conceptual Layers for the Grid

Jeffery proposed to organize grid conceptual services into three layers.\textsuperscript{33} This conceptual model for the grid consists of data/computation, information, and knowledge, as shown in Figure 5. The roles of these three layers are illustrated as follows.

1. The data/computation layer forms the fabric of the grid to provide raw computing power, high-speed bandwidth, and associated data storage in a secure
and auditable way. It is characterized as being able to deal with diverse resource as a single "metacomputer"\(^5\) (virtual computer).

(2) The information layer addresses the way that information is represented, stored, accessed, shared, and maintained in order to provide homogeneous access to heterogeneous distributed information. Here, information is generated not only by databases but also by instruments, sensors, people, computation analysis, and so forth. So we can see that the following functions will be contained to satisfy requests: workflows composition, metadata access, visualization, data management, and instrumentation management.

(3) Just as information is understood as data equipped with meaning, knowledge is understood as information applied to achieve a goal, solve a problem, or enact a decision. In this layer, knowledge will be acquired, used, retrieved, published, and maintained. So the following functions are required: data mining, visualization, simulation, problem solving methods/environments, semantic portals, and so forth.

Each layer represents a context of the previous layer. Multiple interpretations are possible at the junction between each layer. These layers will ultimately provide rich, seamless, and pervasive access to globally distributed heterogeneous resources. The layered model has been proved useful to promote an expansion of services a grid should support.

4.2. A Grid Computing Prototype Embedding ENGM Layers

Based on the above conceptual model, we present a grid-computing prototype that embeds ENGM layers from the implementation point of view as shown in Figure 5. We can also view this prototype as a three-layered model.

(1) Heterogeneous and Distributed Resources consist of different types of resources, such as network bandwidth, computing power, and storage capabilities available from multiple service providers distributed in grid systems.

(2) ENGM provides the services needed to support a common set of applications in a distributed network environment.\(^3^4\) Through ENGM, virtual organizations can enjoy secure, automatic, reliable, and flexible grid applications. It is made up of extensive ENCE, emergent grid common service, and grid pluggable developing kits.

(1) Extensive ENCE. This sublayer includes extensive ENCE low-level function modules, extensive ENCE core service, and grid agent survivable environment. Grid applications, which are characterized by their very high demands for computing, storage, and network bandwidth, are different from common distributed applications. The feature of supporting pluggable functions and services makes us extend its roles easily to meet requirements of dynamically changing environment.

(2) Emergent Grid Common Services. The contribution we are addressing is utilizing the interactions among ecological agents and interactions between agents and their environment to emerge useful services needed by the third-generation grid systems. Obviously, this sublayer is the kernel of grid computing that is responsible for resource location and allocation, authentication, information service, task assignment, and so on. Note that these common services are emerged by those agents and their invoking low-layered services.
Grid Pluggable Developing Kits. This sublayer provides the developing environment, which contains low-level function development, agent creation, distributed computing, remote visualization, collaborative applications, problem solving, and negotiation support. It lets users effectively use grid resources. Note that these toolkits are pluggable in terms of special grid applications.

Grid Applications for Virtual Organization use developing kits and organize certain agents and common services automatically for special purpose applications, such as large-scale distributed collaborative scientific simulation, distributed mission training, virtual surgery for medical instruction, earth observation, and so forth.

4.3. Agents in ENGM

Grid agents and Emergent Grid Common Services invoked by them construct grid services and applications. They fall into two categories: grid user agents and grid service agents. A grid user agent represents a user within the ENGM system. It can be created by users easily through the agent creation toolkit, which is a kind of Grid Pluggable Developing Kit. Grid service agents are used to comprise the main components in ENGM. Each of these kinds of agents follows simple behavior rules and implements a functional component related to its service. Also these agents are organized into one or more dynamic hierarchical grid communities autonomously. In the following, we discuss these key issues of ENGM: dynamic hierarchical grid community, grid user agents, and grid service agents.

(1) The dynamic hierarchical grid community. As described in Section 3, we know that a collection of agents will form a community under a certain law. Emergent functions of communities and supercommunities (higher level communities, described in Section 3.1) provide multilayered grid management with a flexible, easy-to-use, and adaptive approach. For instance, grid user agents may be used to represent peoples’ presence. They manage and represent data about client end (client host). A population of agents distributed on participants’ hosts is grouped as a community. Grid user agents can join or leave the community dynamically. The communities are hierarchical, with one root-level, several middle-level, and many leaf communities (these communities are also regarded as virtual organizations). Several low-level communities will form an upper-level community. Each agent accesses the grid system via a kind of emergent function (called as entrance service). Actually, this kind of dynamic hierarchical grid community is a decentralized logical topological architecture for the grid, which avoids a central point of failure. It is suitable to autonomous systems such as the Internet, which are prerequisite for scalability. Obviously, managing users’ privileges and roles becomes feasible in this way.

(2) Grid user agents. A grid user agent can autonomously provide the personal information such as preferences and conditions of a user to other parts of the system. These agents can help users make some personal decisions, especially when a choice of services becomes available. The decision should not be made arbitrarily, but depends on the preferences and circumstances of a particular user.
For instance, a user may prefer one service to another, or the user’s operating system may only support some forms of interaction between services and the user. The user should not have to query which service must be chosen each time, as these preferences and previous choices can be recorded and acted upon by the user agent to select from each set of options presented to it.

Therefore, it is necessary for user agents to adapt not only dynamic grid circumstances such as network connection, but also behaviors and requirements of users. As discussed in Section 3.1, each agent follows simple behavior rules and has the described knowledge related to its service or functional component such as (but not limited to) recorded data about user preferences. Moreover, these agents will evolve to desirable behaviors and functions to adapt dynamic requirements and circumstances through performing some operations (behaviors) such as migrate, inherit, split, and crossover. Note that even two agents providing the same service still have different and local knowledge eyeshots about its service or functional component. In the other words, every agent has not sufficient knowledge to adapt all possible requirements and dynamic circumstances. To provide better services for returning more energy, it is necessary for these agents that have related services or functional components to share and utilize their knowledge. To achieve these adaptive results, operations performed by user agents will: (1) abandon unnecessary knowledge, (2) split different categories of knowledge, and (3) integrate similar or complementary knowledge to realize adaptation to heterogeneous requirements and dynamic circumstances.

First, it is necessary for knowledge to be deposited into different categories in terms of analysis about grid circumstances and user profiles built on the recorded information such as user requirements, preferences, and previous choices to services. For example, for one special service, knowledge of a user agent will be classified into important, related, and useless. Second, user agents will gain adaptation to available knowledge through performing some operations autonomously under an energy-driven mechanism. In Figure 6, inheritance, split, and crossover operations of user agents are indicated. Obviously, these operations realize abandonment, split, and integration of knowledge categories of a, b, and c. Finally, grid user agents utilize knowledge adjusted by biological operations to learn and serve for satisfying desirable requirements and dynamic circumstances. What is more, through knowledge learning, grid user agents could provide further forecast information to construct more useful services and applications.

In addition, a grid user agent still acts as an intermediary between a user and services within ENGM. Sometimes, services send messages to users, for instance, reminding users to operate data or noticing users changes to remote databases. Users can delegate the details of a procedure to grid user agents.

(3) Grid service agents. In ENGM, main service components are comprised of a collection of multiple grid service agents that have functionality related to their services and follow simple behavior rules similar to ecological entities. Each grid service agent is viewed as a meta-level component of a grid service in ENGM. Extensive ENCE will cooperate with the community formed by a collection of autonomous and self-organizing grid service agents to emerge some Grid Common Services. In the following, we will introduce some kinds of grid service agents.
that construct key grid services, such as Grid Resource Management Agent, Grid Information Service Agent, Remote Data Access Agent, and Grid Security Service Agent.

(1) Grid resource management agent (GRMA). Grid resources are characterized as providing high performance computing capabilities for grid users. Resource management service is the component that performs efficient scheduling of application utilizing available resource in the grid environment. It is of importance to realize grid vision providing seamless, pervasive, and secure use of grid resources. GRMA acts as a representative for a local grid resource at a meta-level of resource management. This “meta” resource that could be an instrument, a PC, a workstation, or a cluster of computers is scheduled by GRMA. This means that an agent can therefore be considered as a service provider of high performance computing capabilities. These kind of agents cooperate to provide the basic services, such as resource advertisement and discovery (resource sensing), with which to manage and schedule applications over available grid resources. Also GRMA may unite common resources and provide a common user interface so that users can finish a task with any common resource. GRMA is a general, ubiquitous service, with specific application toolkit commands built on its top.

(2) Grid information service agent (GISA). It contains static and dynamic grid resource and service information, which is described separately with resource and service description languages such as MDS (Metacomputing Directory Service) and WSDL (Web Service Description Language). It uses the community niche sensing service to

Figure 6. Inherit, split, and crossover operations of user agents.
provide information directory service in the community niche. Note that GRMA also returns the updated information regarding the capabilities and availability of the computing resources to GISA. If a grid user agent with a user’s requests wants to discover a given resource or service, it forwards this query message to GISA in its community niche. If the query target is found, then the querying is the end. Otherwise, GISA forwards the query message to another GISA in an up-layer community to look for the required resource or service. But for users, the operation is very simple. For instance, if a user wants to start a service of the ENGM system while his own computer is busy, he only needs to input his request as to which computer is free in a community (e.g., DHU-COMMUNITY) in ENGM uniform GUI. And then a grid user agent with his request forwards this query message to GISA. The GISA works with a command query-computer -lim DHU-COMMUNITY. If the query target is found, this user will get the answer that YSDING.DHU.EDU.CN is free in ENGM GUI. After that, the user makes a request for using this computer.

3) Remote data access agent (RDAA). One characteristic of the grid is that large data collections geographically distributed are emerging as important community resources. Users, especially the communities of researchers, need to access and analyze these kinds of data collections. RDAA is used to help users access remote data. It supports some remote access operations such as remote data read and write.

4) Grid security service agent (GSSA). Security issues such as authentication, authorization, and accounting mechanisms need to be set in place and function in the context of increasing scale and automation. Public key infrastructure is introduced into the grid security service (GSS) of ENGM. A group of agents named GSSA cooperate and invoke an Extensive ENCE Security authentication service to provide GSS in each community niche. GSS acts as CA (Certificate Authority) and generates the public key and private key for the users. It keeps owned users’ public key and some private information about users, such as a password, which is used for identifying the user in ordinary ways. Users can use an authentication ticket generated by GSSA to access resources, request services, and so on.

4.4. Observation on ENGM

The above dynamic hierarchical grid community provides a decentralized logical topological architecture for the grid. The services such as resource management, information service, remote data access service, and the security service are result from the interactions among ENGM grid service agents and interactions between agents and their environment. ENGM achieves built-in mechanisms to support some key features of the natural ecosystem such as scalability, adaptability, self-evolution, self-organization, simplicity, and survivability favorable for solving issues that next-generation grid systems face. In the following, we mainly show three of them.

1) Scalability. Here, scalability refers to not only the scalability of the wide-area distributed computing infrastructure but also of services and applications. Centralized approaches have scalability problems as there is one central authority coordinating the activities. And the dynamic hierarchical grid community is already directed to a scalable solution. It is a decentralized logical topological architecture for the grid. Agents, including grid user agents and grid service agents, can join or leave the community dynamically. What is more, a community is scalable because all of its agents act autonomously and locally based on local information in their environments. There is no master agent that controls all agents in a community, so
the number of these agents will grow freely for gaining more energy benefits. When the size and population of a community increases, each agent in this community still repeats the same acts as before. Therefore, a service or an application emerged by a community (or communities) will share a scalability feature.

(2) Adaptability. The communities or higher level communities adapt to the heterogeneous and dynamic grid environment and requirements of users through the emergent behaviors and interactions of agents in them. Agents will evolve to desirable behaviors and functions to adapt dynamic requirements and circumstances through performing some operations (to their related information) or behaviors (to themselves). In Section 4.3, some operations about abandonment, split, and integration of knowledge were introduced to realize adaptation to heterogeneous requirements and dynamic circumstances. In addition, agents will autonomously perform some behaviors to reach adaptability; for instance, they may migrate under an energy-driven mechanism toward the source of requests for avoiding areas of the grid where resource costs are high. At the same time, agents are designed to replicate when demand is high and to die when demand is low. There are many other approaches proposed for improving the adaptability of grid services and applications in ENGM. For instance, each agent defines one or more interaction modes that are designed as food web-based policy. When the grid environment changes, for example, when there are insufficient resources or service requests, the interactions of some agents will change from “cooperation” to “competition” for adapting to the changes. Also interaction control service provides some selective strategies for choosing “appropriate” agents and interaction modes.

(3) Self-evolution. The realization for evolution of services is natural selection from diversity of services. Diverse grid services are emerged by autonomously formed communities. As to different laws, agents may make up diverse functional communities. There are diverse service or functional component of agents (created through their behaviors or operations described above), which also contribute to the diversity of grid service in ENGM. Because energy-driven control service as a natural selection mechanism chooses useful and survivable services from diverse grid services, agents can evolve to desirable behaviors and functions to adapt dynamic requirements and circumstances.

5. AN APPLICATION DEMONSTRATION

Grid is also viewed as the next generation information technology (IT) infrastructure, enabling virtualization of heterogeneous and distributed resources and connecting enterprises in a great global grid. The emergence of the commercial grid is characterized by more dynamic, nimble, and flexible business practices where new supply chain managements are built in response to constantly changing and increasing customized market demands. Success in today’s business world requires that enterprises have the ability to rapidly analyze the value of a new business opportunity, dynamically identify potential partners to respond to them, effectively increase operational flexibility involving improving collaboration among employees, partners, and suppliers, and deftly delivering goods or services across the resulting value chain.
An ENGM platform-based commercial grid service environment can be set up on some virtual nodes (such as a plant or a set of shops in a plant) in a physical grid, as shown in Figure 7. We have developed a prototype of an enterprise supply chain management system. Figure 8 shows the architecture of the supply chain system that is developed with ENGM for coordinated supply chain planning, scheduling, partner selection, and so on.

5.1. The Architecture of an ENGM-Based Supply Chain System

This architecture aims at providing a framework for dynamic identification of supply chain partners and coordinated development, evaluation, and manipulation of planning/scheduling solutions at multiple levels of abstraction across the supply
chain. Four layers are constructed and each layer has its corresponding functions. They are the Commercial Grid Resources Layer, the Commercial Grid Service Agents Layer, the Commercial Grid User Agents Layer, and the End User Layer.

All kinds of distributed commercial resources are included in the Commercial Grid Resources Layer, such as computing power, databases, models, and so on. These resources could be accessed by agents. For example, the public information about a supply chain entity (such as an enterprise) is stored in a database, which is managed by an Oracle DBMS (Database Management System). The database can be accessed by applications (agents) through standard SQL (Structure Query Language) sentences.

The Commercial Grid Service Agents Layer and the Commercial Grid User Agents Layer are both built on the ENGM platform, which is the core part of the system.

The Commercial Grid Service Agents Layer is the base of the ENGM platform. GSSA, CRMA, GISA, and RDAA are included in this layer. In addition, this layer is still constructed by many other kinds of agents and each agent has its particular functions for a given commercial application, such as a Bargain Agent. A Bargain Agent can employ different strategies to bargain with another buyer agent, so it can act as a seller agent.

A commercial grid user agent in the Commercial Grid User Agents Layer can represent a user or a kind of user task within the ENGM system. When a user task is received by the ENGM platform hosted in a grid node, a commercial grid user agent wrapped with user personal information such as preferences and requirements for tasks will be created. This agent will be dissolved when the task is finished and the results are sent to the end user. Supplier agents are a kind of special commercial grid user agent. Each supplier agent is responsible for supporting the development and revision of planning/scheduling solutions for one or more supply chain entities at a particular level of abstraction. In a dynamic hierarchical grid agent community, low-level agents are typically wrappers for planning/scheduling components that support individual facilities over short- to medium-term horizons. Higher level agents are coordination wrappers for tactical or strategic planning/scheduling components that generally require looking over longer horizons and across multiple facilities (typically within the same company). The Commercial Grid User Agents Layer illustrates the interaction between various types of multilevel agents.

The End Users Layer includes a set of individual users and/or enterprise users who are allowed to share and access grid resources. This layer establishes a friendly Web-based interface between users and application systems. In this layer, many services are provided through web pages and can be implemented by Java Bean, JSP, Servlet, and so on.

5.2. Empirical Evaluation of the Supply Chain Configuration and Reconfiguration Based on the ENGM Platform

The above proposed architecture is used to evaluate supply chain configuration and reconfiguration. The supply chain model is the one we used in Ref. 38.
In the proposed architecture, supplier agents can be dynamically configured and reconfigured to accommodate the introduction of, for example, new products, new product flows, new facilities, new suppliers, and new transportation arrangements. Explicit product, process, and facility models based on energy-driven mechanism will determine the agents with which to coordinate. Changes in these models, such as the identification of a new supplier for a particular component or a new product flow, trigger the automatic recalculation of relationships between supplier agents in the architecture. These agents follow simple behavior rules and have the easy integration of multiple planning, scheduling, partner selection, coordination, and analysis components. Within a given agent, these components can be activated to develop, evaluate, and revise integrated planning/scheduling solutions and associated supply chain arrangements. Solutions are stored in a data structure accessible to all the agents’ components. It is also necessary for these supplier agents that have related services or functional components to share and study their knowledge.

Supply chain configuration is, informally, the problem of assembling a collection of agents that, given local knowledge and communication, can transform basic products into composite products of value. In our model, we use the term “goods” to refer to any discrete resource or task. Figure 9 shows an example of the original configuration of the supply chain before optimization. We can realize the model in an ENGM platform-based commercial grid service environment and optimize its cost. $a_i$ represents a supply chain entity, $g_i$ represents the goods which are supplied by this supply chain entity, $C_{ij}$ represents the fixed cost of $g_i$ ($i = 1, 2, \ldots, n; j = 1, 2, \ldots, m$), $C'_{ij}$ represents the additional cost of $g_i$. Both $a_1$ and $a_2$ can provide $a_3$ with $g_1$ goods. The goods $g_2$ and $g_3$ required by $a_5$ are, respectively, supplied by $a_3$ and $a_4$. It is important to select the best goods for each entity and optimize the whole supply chain considering the prices and quality of the goods.

For experimenting on the ENGM Platform, we formulate the supply chain configuration problem as follows. A case of the unoptimized OR/AND graph is first created (see Figure 10). Each possible configuration of the supply chain is reviewed to verify one whose cost is minimal. The cost of each entity is the sum of $C'$ and $C$, and it may be changed through varying graph with nodes and links.

![Figure 9. OR/AND graph before optimization.](image-url)
Various goods provided to one supply chain entity are supplied by multiple entities. These commercial grid user agents representing multiple kinds of goods interact and configure a supply chain. A link’s cost represents the additional cost, and the bigger value means the higher cost. Based on an energy-driven mechanism, supplier agents invoke Emergent Grid Common Services provided by ENGM, coordinate with other agents, and evaluate integrated planning/scheduling solutions. The dynamic hierarchical grid agent community and energy-driven mechanism determine this kind of autonomous supply chain configuration.

Here, some terms are defined:

- goods \((i, j)\): the additional cost of goods \(i\) provided by the supplier \(j\),
- agentlist\((i, j)\): the suppliers’ code names on the supply chain after optimization,
- goodslist\((i, j)\): the goods’ code names on the supply chain after optimization,
- \(P\): the total costs on the supply chain after optimization.

After optimization, we obtained the results expressed by matrix as follows:

- \(\text{agentlist}(i, j) = [9 \ 6 \ 1 \ 8 \ 5]^T\),
- \(\text{goodslist}(i, j) = [5 \ 1 \ 6 \ 4]^T\),
- \(C = 32.50\).

Viewed from experimental results, the supply chain configured by supplier \(a_1, a_5, a_6, a_8,\) and \(a_9\) is the best one, which means that the supply chain can be configured through the ENGM platform. After the supply chain configuration, changes in the consumer market could cause the fluctuation of consumer demands, changes of the raw material costs, the emergences of enterprise “zero” stock, and so on. The supply chain should response to these changes. In such cases, the supply chain could autonomously add new entities or delete the unsuitable entities to adapt their relationships so that a supply chain is reconfigured. So the supply chain
reconfiguration means that the supply chain dynamically alters itself according to the inner or outer environment’s changes. For example, the change of additional cost, such as goods (6, 8), will lead to the reconfiguration of the supply chain and the changes of agentlist and goodlist. The experimental results are shown in Table I. In the previous results, the supply chain entity $a_8$ supplies $a_9$ with goods $g_6$. The additional cost will increase because the quality of goods declines or the transport costs rise suddenly. At this time, the supply chain will reconfigure and new experimental results will emerge. The previous chain changes and $a_8$ loses the competitive advantage because of the poor environment. The supply chain deletes $a_8$ to pursue the entire benefit (i.e., to minimize the total cost) and $a_9$ selects $a_7$ to supply goods $a_6$. The total cost is steady and it does not rise with the bad environment of supplier $a_8$.

Through the above experiments on the ENGM platform, we find the proposed ENGM-based supply chain system could satisfy various enterprise users’ requirements and gain the best solution to supply chain configuration and reconfiguration. During the process of dynamic identification of supply chain partners and coordinated development, evaluation, and manipulation of planning/scheduling solutions, the proposed system exhibits some desirable requirements of new grid systems, such as scalability, adaptability, and self-evolution. What is more, it would be straightforward to extend the model in various ways, such as providing for consumer demand for multiple goods, alternative producer technologies, and multiple output choices. We expect that such extensions are often computationally tractable for problems of realistic size in most business-to-business e-commerce markets.

6. CONCLUSIONS

In this article, a novel grid middleware ENGM based on ENCE is proposed to address the challenges of the third-generation grid systems. The ENGM that couples ecological mechanisms with agent approaches is competent for desirable requirements of new grid systems such as survivability, simplicity, scalability, adaptability, self-organization, and dynamic configuration and reconfiguration. We also built an ENGM platform-based commercial grid service environment and developed a prototype of an enterprise supply chain management system.

<p>| Table I. The empirical evaluation result of the reconfiguration after change of additional cost. |
|---|---|---|---|---|</p>
<table>
<thead>
<tr>
<th>S</th>
<th>Goods (6, 8)</th>
<th>P</th>
<th>Agentlist</th>
<th>goodlist</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3.1</td>
<td>32.60</td>
<td>[9 6 1 8 5]T</td>
<td>[5 1 6 4]T</td>
</tr>
<tr>
<td>2</td>
<td>3.2</td>
<td>32.70</td>
<td>[9 6 1 8 5]T</td>
<td>[5 1 6 4]T</td>
</tr>
<tr>
<td>3</td>
<td>3.4</td>
<td>32.90</td>
<td>[9 6 1 8 5]T</td>
<td>[5 1 6 4]T</td>
</tr>
<tr>
<td>4</td>
<td>3.6</td>
<td>33.00</td>
<td>[9 6 1 7 3 4]T</td>
<td>[5 1 6 2 3]T</td>
</tr>
<tr>
<td>5</td>
<td>3.8</td>
<td>33.00</td>
<td>[9 6 1 7 3 4]T</td>
<td>[5 1 6 2 3]T</td>
</tr>
<tr>
<td>6</td>
<td>4.0</td>
<td>33.00</td>
<td>[9 6 1 7 3 4]T</td>
<td>[5 1 6 2 3]T</td>
</tr>
</tbody>
</table>
The experimental results demonstrate that the proposed ENGM is suitable for new generation grid applications.

It should be noted that the proposed ENGM will support and be integrated with other grid middleware, for example, Globus, to maintain links with development in the grid community. To realize grid vision, it is desirable to incorporate existing or proprietary components and technologies, and to assemble these components and technologies in a flexible manner.\textsuperscript{1,3,35} Considering interoperability of ENGM and other grid middleware, the definition of the functional interface is very important. For agents, they need to provide their interfaces of runtime services that invoke the middleware. These interfaces can be loaded dynamically, and when an agent migrates to another platform, a platform-specific interface implementation can be bound to the agent. For the middleware, the components that implement the required functionality should have a clear interface and could be replaced by other implementations.

The next work is on issues about integration of ENGM and Globus. Also, more experiments will be designed to evaluate the ENGM and its approaches in a grid environment.

Acknowledgments

This research was supported in part by the National Natural Science Foundation (60004006), Shanghai Youth Scientific Technology Qimingxing Project (00QD14038), and Shanghai Youth Teacher Foundation (01QN33) from P.R. China.

References


