ISSN 1064-2307, Journal of Computer and Systems Sciences International, 2012, Vol. 51, No. 2, pp. 256–281. © Pleiades Publishing, Ltd., 2012.
Original Russian Text © V.I. Gorodetskii, 2012, published in Izvestiya Akademii Nauk. Teoriya i Sistemy Upravleniya, 2012, No. 2, pp. 92–120.

COMPUTER METHODS

Self-Organization and Multiagent Systems: I. Models of Multiagent Self-Organization

V. I. Gorodetskii

St. Petersburg Institute for Information Science and Automation, Russian Academy of Sciences, 14 liniya 39, St. Petersburg, Russia

e-mail: gor@mail.iias.spb.su

Received December 7, 2010; in final form, June 7, 2011

Abstract—Nowadays, there are many problems whose complexity is much higher than the capabilities of modern information technologies. Such problems arise in economics, ecology, managing state-level infrastructures and global computer and telecommunication systems, ensuring the safety of society, and in many other fields. Even though these problems seem to be quite different, they have many common features, which imply common difficulties in their solution. These features are as follows: they are large-scale problems, they are open, have unpredictable dynamics and complex structure, include mobile components, and some others. The management in such systems is a challenging task, which requires a revision of modern views, models, architectures, and development technologies. A response to this challenge is the increasing activity in the field of principles and mechanisms of self-organization and in the software tools for their development. Although the paradigm of self-organizing control systems is not new, it is now at a new step of development, which involves, in particular, its integration with the multiagent system paradigm. The purposes of this paper are to analyze the state-of-the art in the field of multiagent self-organizing systems, to provide a critical review of the available applications, analyze development techniques, and generalize the results obtained in this field. The paper consists of two parts. In the first part, we discuss the modern interpretation of the principles of self-organization is analyzed, and the reasons for which the integration of these principles with the achievements in the field of multiagent systems provides a new impetus to the development of information technologies in the context of most complex modern applications. A systematization and description of the self-organization models and mechanisms implemented in the framework of the multiagent architecture is given, and biological self-organization mechanisms are discussed. Applications of self-organizing multiagent systems in telecommunication, grid[1] resource management, and routing in computer networks with dynamic topology, as well as applications in distributed learning and in detecting intrusions into computer networks are described.

DOI: 10.1134/S106423071201008X

INTRODUCTION

The main trends in the development of modern information technologies are in many respects determined by most important practical challenges, which often are of global importance. Such challenges arise in economics, ecology, managing global computer infrastructures, ensuring the safety of society and individuals, and ensuring a high standard of living in a broad sense. In the last decade, due to the economic globalization and ecological problems, increased level and qualitative change in safety threats, and because of increasing differences in the living standards in different countries, these challenges become more complicated and, on the other hand, their importance increases. In this context, the achievements in computing and network technologies provide new possibilities but also advance new demands for information technologies.

The problems mentioned above and software systems used for their solution seem to be very different; however, they have many common features, which actually determine the requirements for modern information technologies and determine the trends in their development. Among these common features, a most important one is that the problems are large-scale. Typically, such systems consist of a huge number of *autonomous* entities, which can amount to thousands, millions, and even more entities. Often, these entities are distributed over space and form a network structure; they may pursue their own local goals,

[1] Grid is a virtual (geographically) distributed supercomputer consisting of loosely coupled heterogeneous networked computers that together perform a flow of tasks. Grid computing is a form of distributed computing on a grid in which the selection of a particular computer to perform a task is made by the grid management system.

which may differ from the global goals of the whole system; however, they have to interact or are interested in interaction for various reasons. The software implementing such systems must be *open* because the constituent autonomous entities may leave the system or enter it at any time, and the links between them may change; thus such network systems are *dynamic* both in their composition and topology. The individual autonomous entities have only very *limited knowledge* about the other components of the network, about their goals and capabilities, and about the network topology as a whole. Typically, such systems operate in a *dynamic environment*, and each entity can sense only a limited set of characteristics of its close neighborhood. Other specific features occur when the system components (the constituent autonomous entities) are *mobile* and have a limited communication range. It is unrealistic to implement optimal control (far less, centralized optimal control) of such systems. This is unrealistic not only because of the huge size of the control problem, the unpredictability of its dynamic structure, the dynamics of the environment, and the diversity of the goals of the constituent entities, but also for organizational reasons; indeed, the system may lack a common control center. For example, the systems based on the Internet have no common control center.

Under these conditions, it is clear that a completely new approach to control methods and architecture of complex systems should be used and new information technologies should be developed. The practical implementation of the systems mentioned above is a challenge, which requires many modern views, methods, models, and design and software development tools to be revised. The comprehension of this fact is reflected, in particular, in the new initiative in the field of information technologies formulated in the common project of the G8 countries called *Interdisciplinary Program on Application Software towards Exascale Computing for Global Issues* [1]. This project assumes joint effort of researchers from different countries to develop a new paradigm for designing software and creating infrastructure and software development tools and exascale computer systems able to effectively solve global issues.

A modern response to this challenge is considerable intensification of research concerning the principles, models, and mechanisms of self-organization, as well as software designed for developing self-organizing systems. Although the term *self-organization* was already used by Descartes, in cybernetics the concept of a self-organizing system was introduced by Ashby [2], who defines it as a system able to modify its own structure. Somewhat later, the concept of self-organization as a means of interaction of a system's elements via the environment was introduced by Grassé [3], who studied the social behavior in colonies of insects. In particular, he studied the coordination mechanisms in termites' behavior when they build a nest. For such a type of self-organization, he introduced the term *stigmergy*. Presently, this term is used for the description the self-organization mechanisms in which the autonomous entities composing the system do not interact directly.

In 1959, the first interdisciplinary conference on self-organization was held in the USA [4]. It collected almost all leading researchers in cybernetics of that time, among them Rosenblatt, McCulloch, Newell, Simon, and others; later, they became world-wide authorities in artificial intelligence. At this conference, fundamental problems in the field of self-organization were formulated, but their solution became possible only much later. The problems discussed at this conference mainly concerned machine learning, automata behavior, logical models of problem solving, selection, and so on. Much effort was focused on feedback from the environment. The importance of studying the principles of reliable operation of biological systems and neuron-like structures was emphasized, as well as the importance of network organization of components of self-organizing systems. The subject of this discussion was well ahead of its time. For example, in Foerster's paper, some ideas concerning the thermodynamic concept of self-organization were presented, even though in an implicit form. This concept was clearly formulated by Prigogine 17 years later (see [5]) as the process of emerging order out of chaos in an open system when it absorbs energy from the outside.

It is worth noting that self-organization as a phenomenon in its own right is studied not only in cybernetics but also in physics, chemistry, biology, ecology, economics, artificial systems, and so on. The ideas and directions of research in these fields are described, for example, in [6]. For a long time, self-organization and self-organizing system did not attract much attention. A surge of interest in such systems began in the end of the 1990s due to the emergence of applications where such a model was very attractive especially in combination with multiagent systems (MASs). In the last decades, the interest in self-organizing MASs has been rapidly increasing, and by now a separate promising research and application field in information technologies has formed although very recently these studies were uncoordinated and concerned only successful attempts at conceptualization and algorithmization of specific multiagent applications.

A significant attempt at the generalization of results and structuring the research in this field was made in the European AgentLink project,[2] where a working group for studying self-organizing MASs was cre-

[2] http://www.agentlink.org/index.php.

ated. This group in many respects determined the modern directions of research in the field of self-organizing MASs and gave impetus for its further development.

The theory and practice of self-organizing MASs is nowadays at the stage of making and rapid development; some results have already found an application. Several dedicated volumes of collected articles have been published by Springer (LNAI 2977, LNAI 3910, LNAI 5918, and others). International workshops, seminars, and conferences devoted to this topic are organized annually. The set of possible applications of self-organizing MASs is increasing. In particular, this model seems to be promising in such application classes as security of computer networks, grid systems management, distributed planning and scheduling in transportation logistics, managing complex production facilities, managing sensor networks, mobile computer and communication systems, managing electric networks, information processing, advising in intelligent space models, and many others. As a special subfield, the technology and tools for the development of self-organizing MASs evolves (see [7–15]).

The purpose of this paper is to provide an introduction to the state-of-the-art in the field of self-organizing MASs, a critical survey and generalization of the basic results, including those obtained by the author, systematization of self-organization models and mechanisms implemented in MAS architectures, and give a brief description of numerous applications developed with the use of self-organization principles.

1. SELF-ORGANIZATION, EMERGENCE, AND MULTIAGENT SYSTEMS

1.1. The Concept of Self-Organization and Its Properties

The term *self-organization* is composed of two words—*self*, which emphasizes the basic internal motive (typically distributed) processes of local interactions between the system's components, and *organization*, which is interpreted as the set of relations between the components determining both a structure on the set of components and their interaction in the course of operation.

According to a conventional definition, self-organization is a mechanism or a process enabling a system to change its organization without explicit external control during its operation [16]. It is emphasized in this definition that self-organization emerges and changes only due to internal interactions.

A self-organizing system in which there is no explicit external or internal centralized control is said to be *strong*. If a system has an explicit internal (central) control, it is said to be *weak* [16]. In more detail, self-organizing systems are determined by the following set of properties [16]:

Autonomy, that is, absence of external control.

Global order (organization, structure), which appears in the system due to internal local interactions between its components.

Emergence, that is, properties that are observed only at the global level but cannot be deduced from examining the individual behavior of the system components.

Dissipation, which is a kind of dissipation of some "energy" in the unstable states of the system in the absence of external perturbations, which causes the system to go to certain stable states, in which its emergence properties are observed.

Nonlinear dynamics, instability, and *sensitivity* to the initial states and small variations of the parameters. As a result, small changes in the initial state and the parameters at certain critical points of the state space can cause significant changes in the system behavior; furthermore, this property cannot be understood by examining the behavior of the individual components and their interactions.

Multiplicity of stable states (they are called attractors).

Redundancy of components and their interactions, which makes the system insensible to local damage of its components (fault tolerance).

Adaptation, which is the ability to modify its behavior and go to a new stable state as a response to changes in the organization of the system's environment; in this case, the system behavior often changes in a jump.

Complexity, which arises due to the fact that self-organizing systems typically consist of a large number of components and their global properties and behavior are irreducible to a combination of the properties of individual components.

Simple component interaction rules, which result in the complex behavior of the system as a whole; furthermore, this behavior is not implied by the description of the interaction rules.

Hierarchical pattern: a self-organizing system is described at least on two hierarchical levels—the local component interaction level and the metalevel on which its emergence properties are described.

In future, the properties of self-organizing systems can be extended by including self-diagnostics, self-recovery, self-reproduction, and other properties (see [17]). Notice that, if a self-organizing system is con-

Fig. 1. School of fish. An example of combined manifestation of the properties of emergence and self-organization (borrowed from [19]).

sidered as a distributed network system, then its organization can be either described explicitly or emerge ad-hoc. We also emphasize that the individual interacting components of the system must be distinguishable.

1.2. Self-Organization and Emergence

The concept of emergence has no unambiguous definition in the literature. Typically, emergence is defined as the appearance of a global order, structure, property, or another pattern on the macrolevel as an integral result of the local interactions of the system elements. An example of emergence in physics is temperature, which arises from the chaotic motion of molecules at an average speed. Another example is given by convection, which corresponds to the statistical ordering of molecule motion according to the temperature gradient. One more example is the magnetic field as a result of a coherent spin orientation. An example from biology is the visible motion of a flock of birds or a school of fish (see Fig. 1), which is formed from the motion of the individuals whose interactions obey certain rules.

The following properties of emergence are usually described as the basic ones:

Novelty, which implies that the phenomenon cannot be described in terms of the microlevel components.

Coherence, which is a consequence of local interactions.

Macrolevel manifestation (with respect to the generating components).

Dynamics, which emerges in the course of interactions but is not specified in advance or from without.

Robustness, which is the ability to maintain the system's state under external and internal disturbances.

Ostensivity, which is the ability to explicitly demonstrate itself in a certain way.

The concept of emergence is not always related to the concept of self-organization. Most authors consider these concepts as independent ones (e.g., see [18]). For example, the motion of a school of fish is a self-organizing process with emergence. It can be observed visually and emerges as a result of adhering to some local rules, which are reduced to maintaining an average speed and direction by all the participants of the motion and to maintaining a certain distance between the individuals (Fig. 1). In the case of temperature and magnetic field, we have only the emergence phenomenon. An example of a self-organizing system that has no emergent properties is the system for forecasting floods on the Rhone; this example is considered in the second part of the present work.

The main similarity in the concepts of emergence and self-organization is that both are dynamic processes, which are characterized by the increase in order, are caused by local interactions on the microlevel, and manifest themselves on the macrolevel. The difference in these concepts (see [18]) is that emergence is robust with respect to the set of components whose interaction causes this phenomenon (some components may appear or disappear, but the emergence pattern persists); in distinction, self-organization is an adaptive process. Notice that adaptation manifests itself in that the system goes to a new stable set by changing its behavior and organization when the system and (or) the environment undergo certain changes.

The practice shows that the coexistence of self-organization and emergence in a complex system is quite natural. Self-organization enables a complex system consisting of a large number of simple elements

that have limited information to adapt itself to the dynamic environment based only on the local interactions of its elements. The interactions must be specified in such a way as to ensure the desired emergence properties of the system as a whole, for example, to optimize its most important quality.

1.3. Self-Organization and Multiagent Systems

The software of most prototypes of self-organizing systems developed by the present time is implemented in the MAS architecture, which is quite natural. Indeed, the main requirements for the software implementation of a self-organizing system are as follows:

Its components must be autonomous; that is, they must be able to control their own behavior aimed at achieving local goals without external intervention.

They must be able to perceive the external world and locally affect it.

They must have a software or physical environment for distributed interaction.

They must be able to maintain the system organization (relationships between different autonomous components) and have means for the behavior coordination.

Presently, MAS is the only paradigm that has all the necessary means for fulfilling the above requirements for self-organizing systems. For that reason, the main development of the principles, methods, and models of self-organization as well as their software implementation goes in the framework of MASs, as has been mentioned in the Introduction.

2. CLASSIFICATION OF SELF-ORGANIZATION MECHANISMS IN AGENT-BASED SYSTEMS

Presently, there is no conventional classification of the self-organization mechanisms. Different researchers use different features for such a classification. Some of the proposed classifications do not make it possible to unambiguously assign each mechanism to one particular class. The reason is that many mechanisms are borrowed from biology, where the self-organization processes are complicated and many of them combine several principles. For that reason, the classification principles and the corresponding classifications described below should only considered as an attempt at their systematization. In [19], taxonomy of self-organization mechanisms is formed as a result of answering three questions. We now describe this taxonomy.

1. *Which information is exchanged when local entities (agents) interact?* Two types of such information are distinguished. The first one includes the so-called markers. The interaction based on markers is characterized by the fact that the agents explicitly exchange symbols, signals, and the like having special meaning; the semantics of these signals is identically interpreted by all the agents. These can be voice signals (for example, danger voice signals in a flock of birds), chemical substances giving out odor (such substances are called pheromones), and the like. The other type of information is the so-called sematectonic information, which is interpreted by the entities more or less unambiguously (most often as a similarity). For example, when insects sort objects by putting them in a certain place, by putting an object next to another an agent implicitly indicates that this is a place for similar objects. This interaction mechanism does not require attaching markers to messages sent in the course of the interaction and it does not require a marker recognition mechanism; therefore, sematectonic interactions are less costly; however, markers provide a more expressive mechanism. The sematectonic mechanism can be transformed into the marker-based mechanism, while the converse is not always possible. Another important difference in these mechanisms is that, in the case of sematectonic interactions, the agents must have a common environment (for example, a visual one) in which the meaning of the information can be directly communicated to the receiving agent. In the case of markers, the information can also be transmitted by seeking addressee with the help of mediators.

2. *What is the flow of information?* The information flow is generated by an agent at one of the environment points, while the other agents may be located at different places in the environment. Therefore, they must seek this information. For example, agents may leave data tuples in the common information space and the other agents must seek these data using, for example, pattern matching techniques. In this case, the message has no specific addressee, and the interaction occurs serendipitously.

Another possibility is the diffusion of information in the environment, when the information is transmitted simultaneously to a large number of agents. An example is broadcasting, when the agents listen to the environment and those of them who here this signal can interact.

3. *How the received information is used?* There are two variants of using the information. In the first one (called trigger-based), the reception of certain data triggers a specific agent activity. This variant is also

called event-driven, and it is one of the standard methods for coordinating the agent behavior. For example, the received information can contain an instruction to go to a position with the specified coordinates, which is properly performed by the recipient. In the second variant (called follow-through), the received information implies a chain of actions of the recipient. For example, the coordinating agent leaves a trail which the recipient agent (or agents) must follow.

However, the majority of researchers use the classification of self-organization mechanisms proposed in [20, 21]. In it, the self-organization mechanisms are classified as follows.

1. *Mechanisms based on direct interaction between agents.* In these mechanisms, the local interactions occur directly between agents. In this case, an organization (structure) of the agents may already exist, and the goal of the interactions is to improve its properties. In a particular case, there may be no structure on the set of agents in the initial state; rather, it can emerge in the process of local interactions between the agents. This model is more often used when a spatial organization must be created and maintained on a set of autonomous agents (see [22]).

2. *Mechanisms based on indirect interaction between agents.* In this case, each autonomous agent affects the environment (changes it), and the other agents perceive these changes and modify their behavior by certain rules based on this information. A typical example of this mechanism is self-organization in an ant colony in seeking food; this example is a biological prototype of a wide class of self-organization mechanisms, which has already been mentioned; it is known as stigmergy. This mechanism is used in many practically important applications.

Both agent interaction mechanisms can be implemented using one of the algorithms described below.

1. *Mechanisms based on reinforcement learning.* In this case, the driving force of the self-organization process is a *utility function* also called *reward function*. The agents try to modify their behavior so as to maximize their reward. The organization emerges and is maintained due to the adaptation of the agents' behavior so as to maximize the reward function. Typically, each agent in such systems has its own local utility function, which determines the reward, and the system as a whole has a global utility function of which the agents are not aware. Typically, a virtual payment is used as the local utility function; this payment depends on the agent's behavior, and it tries to maximize it. The result of the self-organization process is a structuring or factorization of the set of agents, which can lead, for example, to reducing the workload on the communication component (in routing problems), increasing the accuracy of prediction of a certain characteristic of the system's global behavior (e.g., the accuracy of flood forecasting), and the like. If the system operates in a dynamic environment when the topology of the set of agents is dynamic, the agents' behavior changes with time; hence the self-organization is a continuous process of changing the system organization. It was shown in [23] that the mechanisms of this type suit well for distributing tasks between agents and for forming groups of agents performing similar tasks. Generally speaking, the reinforcement-based self-organization mechanisms are also studied in the domain of research called *distributed reinforcement learning* [24].

2. *Cooperation-based mechanisms.* In such systems, self-organization is achieved due to local interactions between agents with the aim to cooperate. The agents must interact in a friendly way demonstrating altruistic behavior. For example, such a mechanism is used in the model proposed in [13, 25]. This mechanism is known as AMAS theory (Adaptive Multi-Agent Systems theory). The self-organizing flood forecasting system discussed in the second part of this paper is a prominent example of this mechanism. In this theory, which aims at solving applied problems using the self-organizing cooperation of agents, local cooperative actions of the agents are used. It is assumed that the agents have certain skills, are able to communicate, have some knowledge about some other agents (neighbors), and have criteria for detecting situations in which conflict resolution requires special agent cooperation mechanisms (they are called *noncooperative situations* (NCS)). Such situations are dangerous for the organization. All the NCSs are classified into three groups:

(1) obscure information is received from the environment,

(2) the received information does not stimulate the agent to perform any actions,

(3) the agent's action is useless for the other agents.

The list of NCSs is determined at design time. When an agent finds itself in such a situation, it tries to return to a cooperative situation in which processes comply with their purpose. This is done using the actions chosen by this agent's self-organization mechanism, which leads to modifications in the system structure. However, all the NCSs must be determined in advance, and the corresponding actions must be planned at design time. In the second part of the present work, this model of the self-organization mechanism will be considered in detail using the flood forecasting system as an example.

3. *Mechanisms based on the use of gradient fields.* Such mechanisms underlie many practically important self-organizing MASs, and they borrow ideas from physics and biology. Examples of gradient fields in

physics are gravitational, electric, and electromagnetic fields. In biology, a prototype is morphogenesis, when the behavior of an organism based on the local interaction of a large number of identically programmed cells (see [26]) is controlled by a chemical substance whose characteristics depend, for example, on the distance to the substance generating source or on the gradient of the substance concentration. In morphogenesis, the chemical substances scattered between the cells control their behavior using local estimates of gradient perception of the scattered proteins, which provide information about the relative (with respect to the substance source) position of the cell and about the direction (morphogen gradients). Cells use various morphogens whose concentration controls the specialization of various metabolic processes.

In agent-based systems that use this self-organization mechanism, an analog of the field and its gradient is a data structure representing the computational (digital) gradient field. In the agents' operating environment, this data structure must be represented in such a form that provides access for other agents (its perception by other agents) at each point of the environment. Examples of such a structure in the form of UML class diagrams and their informal description can be found in [9].

In the general case, the gradient field is described by its name, contextual information (e.g., the location in space and the numerical value of the field strength at this location), and a field propagation rule, which determines the change of the field strength in this space. The field is initiated by a source, which can be a place in the environment, other agents, or other entities of the system. The field carries information about the environment and (or) about the gradient field source. The agents' operation environment can be responsible for the field propagation from the source to the environment agents according to the rule of its variation. The gradient can grow with increasing distance from the source (in this case, it is zero at the initial point) or decrease. The field can be also propagated by the agents. In this case, the agents retranslate the field to their neighbors and modify its strength. The process of translating the field from agent to agent repeats until the field strength becomes lower than a certain threshold; then, it is set to zero. Thus, a shape of the field is formed in the space, which carries context-dependent information needed for coordination. An agent that is located at a particular place perceives parts of the gradient field from its neighbors and selects a behavioral strategy (a deterministic or a random one) controlled by the resulting field. This agent somehow interacts with its neighbors (e.g., moves towards them, sends messages through them, requests information from them) but the source of the coordinating information is the gradient field and its local characteristics that are perceived by each agent. In the general case, the key concepts of the model are as follows (see [27]):

1. Contextual information is represented by a computational field (co-field) spread by agents and or by the infrastructure, and the agents must be able to perceive it.

2. A coordination policy, which enables the agents to use field shape to coordinate their motion (or, more generally, their behavior).

3. Application-specific coordination, which is realized by the agents in the global context.

Consider these aspects of the gradient field in more detail.

A *computational field* (co-field) is represented by a distributed data structure with a unique identifier, a numerical value depending on the location, and a propagation rule in the network determining how the field's numerical value is changing from a node to node. Agents can sense the numerical value of the field and can change it to reflect the local context describing their location and (or) state. In the example of service management in a museum considered in the second part of this paper, co-fields can represent the presence of a visitor and a guide; this information is associated with their location. Using these fields, any visitor can detect a guide location and the distance from it. The gradient of this field indicates the direction to the guide.

A field can be generated and maintained by a special infrastructure and by the network nodes. The network can have a dynamic topology; it can be a peer-to-peer (P2P) network; and so on. Changes of the context in a node induce changes in the field values; the field propagates from node to node (beginning from the neighboring ones) according to a certain rule; thus, the field as a whole is modified.

Coordination policy. The motion coordination (the system's dynamics in the general case) is performed by the field shape and its changes. For example, a museum visitor, having detected a new guide who is closer than the earlier intended one, can change its motion choosing the new guide as the new goal. It is clear that coordination based on co-fields uses only the local context; for that reason, only greedy optimization algorithms can be used in this case. However, this is a drawback of all coordination methods based on local context.

Application-specific coordination. Typically, in order to take into account the specific application context, special fields specifying this context or combinations of different fields whose values are sensed by agents are used. For example, let the guides must be distributed uniformly across the museum. This goal

can be achieved by generating identical co-fields by each of them with the zero value at each guide location. The guides must choose their position at the local minimizers of the total field. It is clear that this kind of coordination can ensure that any number of autonomous entities in the museum meet. Details of this and similar applications are considered in the second part of this paper.

By now, no universal methodology that could help construct a field of the required shape for every particular coordination problem has been proposed. However, the authors of [27] believe that such a methodology can be found. Notice that the biological self-organization mechanisms described below in Sections 3.3–3.8 can be represented in terms of field-based coordination.

4. *Market self-organization mechanisms.* A feature of the market model of the self-organization mechanism is that it considers the distribution of limited resources as a self-organization model under highly dynamic conditions when each agent has access only to local information; therefore, the only possible approach to solving the global problem is decentralized coordination. To implement this approach, it is necessary to somehow associate the available local information about the current use of resources with the global model of their utilization. An appropriate choice of the structure of this information is a key problem that must be solved to hold out hope for the successful application of the market self-organization mechanism. Another, no less important problem, is to know how to form data about the global use of resources when only local information is available.

The origin and the very market self-organization paradigm are borrowed from economics, in particular, from the free market economy. In such an economy, the market participants play either the role of purchasers of goods (in specific applications of self-organizing MASs, goods can be interpreted as tasks, bandwidth, machine tools, and the like) or vendors; both act only in their own interests are *self-interested* in terms of MAS). This metaphor can be effectively used for solving the problem of optimal distribution of goods when the prices converge to globally equilibrium (market) prices: if the quantity of goods is greater than the demand, the prices go down to a new equilibrium; if the quantity of goods is greater than the demand, the prices grow. In the systems that use market-based self-organization mechanisms, the macroeconomic level (equilibrium prices and other properties derived from price information) is formed as a consequence of the emergence arising as a result of local processes on the level of the microeconomic model. Exactly for this reason, the microeconomic level completely borrowed from the economic theory plays the key role in self-organizing systems.

The key problem in the construction of the multiagent model of an application in terms of the self-organizing market is the choice of purchasers and vendors (producers). Various preferences of agents and constraints of the real problem can be introduced in the model algorithmically by selecting the agents' behavior strategy in the course of trading. Another important problem in the construction of the model is matching between the purchasers and vendors that are of interest for each other from the viewpoint of a possible deal. This mechanism must be explicitly described.

Informally, the market self-organization model can be described as an auction in which groups of agents negotiate with the goal to sell or buy scarce resources. The agents interact locally by exchanging messages containing *deal offers, commitments*, and *payments* for resources. It is assumed in this model that the agents possess a certain sum of money at the initial state and each resource is owned or used by an agent at any point in time. In other words, there are no ownerless resources. Some agents have a need for a certain amount of specific resources (these agents are consumers of buyers), and they are ready to pay for them within their budget. Their aim is to pay as little as possible. Other agents act as producers or suppliers; they possess a certain amount of specific resources and are ready to sell them at a certain price. Their aim is to maximize their profit.

In this model, all the agents decide on their own if they are in the role of a buyer or a supplier and participate in transactions with suppliers (respectively, buyers) for buying (respectively, selling) a certain amount of a resource. In this strategy, the behavior of each agent depends on its local information and on the market prices, which play the role of global information. Each agent has a specific minimum and maximum price for each resource unit; this price is not known to the other agents. The buyer cannot pay more than his maximum price, and the vendor cannot sell cheaper than his minimum price. The distribution of the maximum and minimum prices determines the evolution of the curves of the needs and possibilities. In the process of trade, both the minimum and maximum prices evolve in the direction of the market price. Thus, the self-organization mechanism must be constructed in such a way as to ensure that the price can rise and fall dynamically reflecting the relation between the current demand and supply when the supply also changes dynamically. However, the agents set their prices based completely on the implicitly perceived information about the available amount of resources for sell and the demand for it; the agents get this information by offering a specific price.

A special case occurs when an agent is simultaneously a buyer and a vendor. In this case, he must use a strategy that enables him to keep a necessary amount of resource when selling it while trying to maximize his profit.

We have already mentioned that the model of the market self-organization mechanism used in MASs is the model of auction. Many different variants were proposed, which were thoroughly studied theoretically (using the game theory methods) and experimentally. A substantial body of literature is devoted to this line of research (e.g., see [28–33]). A specific feature of the auctions implementing the self-organization model is that they must include, in addition to the conventional set of characteristics and parameters, an application-dependent neighborhood relation on the set of vendor agents and buyer agents. Another variant of the implementation of the self-organization mechanism in MASs is the so-called demand-resource network (DRN) model described in the second part of this paper.

5. *Mechanisms using the holonic system model.* In [21], one more class of self-organization mechanisms is described, which is called the *architecture-based mechanism*. It includes a large class of the so-called *holonic systems*. The basic entity (concept) in this mechanism is holon, which can exist independently, but also can join with other holons thus forming larger holons. The consolidation processes are dynamic, and they form dynamic structures consisting of a hierarchy of holons called holarchies. Holarchies change dynamically depending on changes in the environment; this phenomenon is the essence of the holonic approach to self-organization.

Actually, the theory of holonic systems is an independent direction of research in MAS, which evolves outside the theory of self-organization. For that reason, we do not discuss this self-organization mechanism in the present paper. The concept of holonic systems and the use of this concept as a self-organization mechanism are described in [34].

3. EXAMPLES OF BIOLOGICAL SELF-ORGANIZATION MECHANISMS USED IN MULTIAGENT SYSTEMS

The emergence of the self-organization model and the concept of self-organizing systems are in many respects derived from examples of biological systems. Indeed, the majority of self-organization mechanisms that were proposed and are now practically used in multiagent applications are inspired by biological systems. Typically, these mechanisms are based on fairly simple models, which fit in the classification described above even though some of them use several mechanisms simultaneously.

In this section, we give a short description of some biological self-organization mechanisms. This is done with two goals in mind. The first one is to illustrate by examples the classification described above. The second goal is to facilitate the understanding of how these mechanisms are used in modern multiagent applications including those discussed below.

3.1. Swarm Intelligence

One of the most illustrative mechanisms that inspired a special direction of research in optimization of large-scale systems and in self-organization and made it possible to implement interesting and diverse multiagent applications is self-organization in colonies of insects. Different insect communities use different survival mechanisms. In this subsection, we consider the most well-known mechanism of this group, which is often used in practice—this is the self-organization mechanism used by colonies of ants for the collective search of food and for transporting it to the anthill. This mechanism is usually called swarm intelligence.

Swarm intelligence is based on an indirect interaction mechanism called stigmergy, which was first considered in [3]. In biology, this model works as follows. First, ants wander randomly looking for food. When an ant finds a source of food, it leaves a mark at that place in the form of a chemical substance called pheromone. It also marks by the pheromone its way back to the anthill. When other ants detect a way marked by the pheromone, they choose it with a certain probability. The stronger the pheromone concentration, the greater is the probability that this way is chosen by ants. Each ant that reaches the food source also leaves pheromone there and also marks its way back. Hence, the ways used by a greater number of ants have stronger concentration of pheromone on them thus attracting more ants. However, the pheromone evaporates with time; therefore, when the food source becomes exhausted and the ants do not enhance the concentration of pheromone on the way to it, it does not attract ants any more and disappears in a certain time.

This self-organization mechanism has the following features (see [19]). First, the distributed entities (ants in our example) interact indirectly through the environment by leaving in it their traces, which are sensed by other autonomous entities. The local interactions are multiple and local. Second, this mecha-

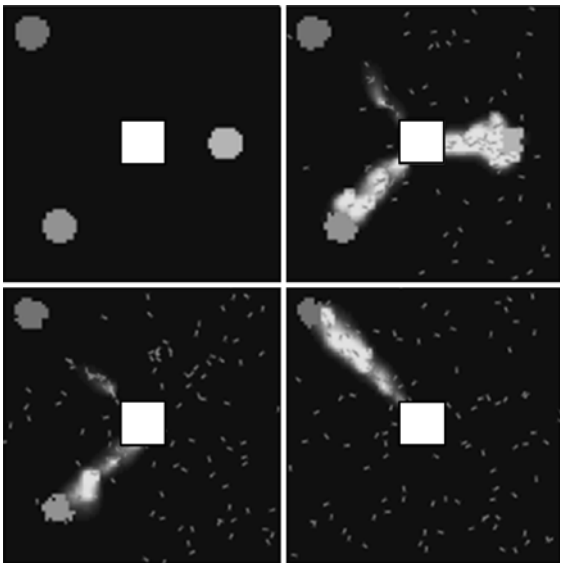

Fig. 2. An example of self-organization based on indirect interactions used by a colony of ants when seeking food and bringing it to the anthill [19].

nism uses a delayed *positive feedback* in the form of increasing the amount of pheromone left by the insects at the point where the food source is detected and on the way to it. Third, this mechanism uses a *negative feedback* realized in the form of pheromone evaporation. Finally, it includes the enhancement of the *behavior modification* as the amount of pheromone on the way to the food source increases.

Figure 2 borrowed from [19] illustrates the dynamics of the self-organization process. The white box indicates the anthill, and the light-colored dots denote the way marked by the pheromone; these dots are the brighter, the stronger is the pheromone concentration on them. The circles indicate food sources. The figure in the left top part corner shows the initial situation when the ants start looking for food by random walking. In some time, when the ants find two food sources and leave enough pheromone for the greater part of the colony to unmistakably find the way to them, we have the situation depicted in the right top part of Fig. 2. Later, when the food in one of the food sources is exhausted but is still left in the other source, we have the situation shown in the left bottom part of Fig. 2. Here, there is enough pheromone on the way to one of the sources while the pheromone on the way to the other source has already evaporated. With time, as this food source runs out and the way to it does not get new pheromone trace, it gradually evaporates. Finally, when the ants find the third food source and fill it with pheromone, we have the situation depicted in the right bottom part of Fig. 2.

In accordance with one of the taxonomies described above, swarm intelligence is classified as a mechanism with the indirect interaction of distributed autonomous entities. According to the other taxonomy, it is based on the use of markers because the pheromone left by an ant is intended to stimulate the interaction, while the interaction itself is realized serendipitously because the ants wander randomly until they meet a pheromone mark. The presence of pheromone in a proper concentration stimulates the ants to follow the pheromone trail.

There are many applications based on the use of the indirect interaction mechanism; some most important of them are described below. In essence, these approaches are based on metaheuristics that are used in the process of local interactions of entities to find a better state of the system as a whole while the entities themselves do not know which state is better. For example, this model underlies some algorithms in the vehicle routing problem, traveling salesman problem, routing packets in telecommunication networks, scheduling problems, and many others. In the general case, this mechanism works well when the problem is reduced to finding the optimal path in a weighted graph.

3.2. Nest Building

The idea of self-organization in the cooperative construction of a nest (e.g., a termitary) in the absence of centralized control from a "master" is implemented using the following very simple local interaction rules (see [17]):

1. Individuals create small clusters ("bricks"), typically of trash, on the surface and mark it with pheromone.

2. They carry those bricks randomly, but prefer to travel to the places with stronger concentration of the pheromone.

3. At each step, the individuals decide stochastically whether to deposit the brick or carry it on.

4. The probability of the brick to be deposited is the greater the stronger the pheromone concentration.

This algorithm leads to the generation of scattered initial deposits. These deposits attract individuals and increase the probability that these individuals will leave a brick there. The most recent deposits are placed at the center of the pile. So, they are the strongest and piles tend to grow vertically rather than horizontally, forming columns. When columns grow near each other, the scent of each attracts individuals of the other, thus pulling subsequent deposits into the shape of an arch. As a result of such a cooperative activity, termitaries as high as 5 meters high and 10 tons in weight are built. These nests are multi-story structures providing storage for food, housing the brood, and protection for the population.

This mechanism is classified as a stigmergy mechanism because it is based on indirect interactions. It is widely used in cooperative robotics, in particular, to coordinate the robot cooperation in assembling various structures.

3.3. Molding

Molding is when individuals are attracted by specialized resources available in the environment. If this resource is plentiful, the individuals wander autonomously and consume it. When the resource becomes scarce, the individuals move toward one another forming a cluster. Then, the cluster begins to crawl about in the environment as a single supercell looking for a more favorable spot. When it finds such a place, the supercell differentiates again, and a new cycle begins.

The mechanism supporting such a type of self-organization is based on the fact that when the individuals are molded in a cluster, they produce a pheromone that diffuses in the neighborhood and attracts other individuals. They follow the gradient of the pheromone, tending to aggregate where the pheromone is the strongest. The molding process is exploited by simple plants and bacteria to survive in harsh environments. Molding is classified as based on sensing the field gradient formed by the pheromone. This kind of mechanism finds practical applications in robotics for robot self-assembly in order to perform some task cooperatively. Examples can be found in [17].

3.4. Morphogenesis

Morphogenesis is an interesting and fundamental self-organization method for a vast range of biological cells. In all its variants, biological species use the gradient formed by a special substance called morphogen, which is used by each cell to determine its position relative to the morphogen source and the direction to it.

This mechanism is utilized by cells in the process of self-organization to get *positional* and *directional information* for the evolution of an embryo. The essence of this mechanism is as follows. Cells at one end of the embryo emit a morphogen that diffuses along the length of the embryo. The concentration of this morphogen decreases with the distance from the source; hence, the concentration can be used as a source of information about the distance of a cell from the initial location of the embryo. For example, using this information, a cell can determine whether it lies in the head, thorax or abdominal region of the embryo.

The growth of the embryo is governed by different morphogens that carry information about the distances from different locations to the current position of the cell that perceives their concentration. For example, the cells in the abdominal region perceive a different morphogen than the cells in the head, and this specific morphogen enables the abdomen cells determine their position within the abdominal region thus controlling the specialization of cells within the abdominal region. The information about the direction is obtained by estimating the morphogen concentration in different directions.

This self-organization mechanism is gradient-based. Its principle can be used in self-organizing MASs that determine the direction of the gradient by collecting information from the neighboring agents and comparing it. This mechanism enables an agent to select a neighbor that is closer to the source and thus iteratively build a route to the source. Note that this principle underlies the self-organization of a road traffic control system described in the second part of the present paper.

One more mechanism used in morphogenesis is the local inhibition and competition that provide a basis for cell differentiation. The essence is that cells emit a special morphogen that inhibits the similar development of the neighbor cells. This principle is a useful primitive for, for example, selecting a leader within a local group of candidate agents. It can be implemented by generating and combining random numbers in a bounded range. The competing agents select random numbers and arrange them in ascend-

ing order. The agent with the minimum value of such a number becomes the leader, and it generates a signal that stops the competition. This simple algorithm is stable under asynchronous behavior, leaving agents, and appearance of new agents in the system.

Morphogenesis includes some more mechanisms that provide interesting primitives for building self-organizing MASs. They include lateral inhibition (when the cells emit an inhibiting morphogen with a small propagation radius, which makes it possible to control the construction of regular spatial structures) and local monitoring in which cells (or agents in applications) periodically poll their neighbors to detect if they are "alive." For that purpose, cells (agents) periodically send messages to indicate that the cell is still alive (or the agent is still in the system). The situation when an entity does not send such a message for a long time is interpreted as the death of this entity (the agent has left the system).

Other biological morhogenetic mechanisms that can be useful for the construction of self-organizing MASs are described in [35]. All these mechanisms use the exchange information in the form of markers, spread information by diffusion, and the received information is used as a trigger for starting the corresponding cell differentiation mechanism.

Presently, morphogenetic mechanisms have wide practical applications. Moreover, many self-organization methods based on gradient fields are akin to morphogenesis, and the self-organization that uses the primitives described above is widely used in, for example, controlling self-assembly of modular robots generating a field as a virtual morphogen (see [17]).

3.5. Web Weaving

A web is woven by spiders to facilitate traveling and catching prey. A variant of web weaving is implemented by spiders as a cooperating self-organizing process.

The web weaving activity is based on the low-level task of connecting two ground spots with a dragline, where a ground point is a place where the web is fastened. As in the other biological mechanisms described above, the individuals first roam randomly weaving a dragline between two random spots. When a dragline is fastened at a point, it forms the new shortest path between two spots (network nodes). Later, the individuals are attracted by already existing draglines and prefer walking upon one of them rather than on the ground. This enhances the chance of having a dragline beginning where an already existing dragline ends. In such a biological system, the probability of walking upon a dragline must be carefully tuned. If it is too low, no web is built and all the available space is filled with disconnected segments. If it is too high, individuals are trapped in their own draglines and no real weaving occurs. Hence, to successfully build a web, a balance between the probability of going along an already existing dragline and moving to a node not on this dragline must be maintained. In accordance with the first taxonomy, this mechanism uses a sematectonic (even though an indirect) interaction. This interaction is implemented serendipitously, and the information obtained through it is involved in the self-organization step by step.

This mechanism, similarly to those discussed above, has found practical applications in self-organizing MASs, for example, as a means for rapidly looking through distributed data in P2P networks and for network routing.

3.6. Brood Sorting

A self-organizing sorting mechanism is used in biological communities for cooperative sorting of eggs, larvae, and the like. In this process, the individuals first wander randomly, take objects to be sorted, and deposit them near similar surrounding objects. For example, if an individual finds a greater object near a pile of similar objects, it most likely takes it and start wandering around and deposits it near similar objects. This is one of the algorithms used by ants.

In accordance with the first taxonomy, this mechanism uses a sematectonic information (size of objects to be sorted), serendipitous interaction (when the individuals randomly find piles of objects), and the response of the individuals to the received information is a trigger-like one (take an object, drop an object). In practice, this mechanism is used for database organization and virus protection. For example, a swarm of simple software agents can look through resources and their clusters, and detect and remove outliers (e.g., infected files).

3.7. Flocking, Schooling and Herding

This self-organizing process is driven by three interaction components: keeping a certain distance between the individuals to avoid collision, maintaining a common speed, and centering, which stimulates each individual to move towards the flock center. This coordinated movement happens without a leader,

and each individual can maneuver at any time. Such a flock never attains equilibrium, and its shape is unpredictable. The attractiveness of a flock to individuals grows with the size of the flock. Every flock includes sentinels. Sentinels are individuals that are more sensitive to such stimuli as sound, smell, or approach of danger, and they react earlier to these stimuli than the other individuals. After a sentinel moves, other individuals tend to follow due to the forces mentioned above.

According to the first taxonomy, flocking is sematectonic, it is based on diffusion in transmitting the (visual) information, and it is follow-through. According to the other taxonomy, this mechanism is based on the direct interaction between the individuals, and it is implemented using local cooperation of the neighbors in the flock. This mechanism can be useful in a large variety of scenarios, for example, for controlling a group of unmanned aerial vehicles.

3.8. Quorum Mechanism

Quorum is the mechanism in which individuals self-organize their behavior according to the behavior of the majority of them. This phenomenon occurs when a minimum number of individuals (critical mass) behave similarly. This phenomenon is driven by the signals that are perceived by the neighbors, and their common signal triggers other individuals to copy their behavior. The signal is assumed to disappear over time; however the quorum still exists for some time, and the rate of its fading typically does not affect the duration of the quorum existence. An example of this mechanism is the cooperative luminescence of lightning bugs.

4. EXAMPLES OF USING SELF-ORGANIZING MULTIAGENT SYSTEMS IN SOFTWARE INFRASTRUCTURES OF COMPUTER NETWORKS

Presently, the most interesting and practically important applications of the idea of self-organization are in the field of software infrastructures of computer networks and open systems with dynamic architecture. The reason is clear—the complexity of the problems arising in these applications by far exceeds the capability of the conventional control and optimization methods. We consider some of the most productive variants of using self-organizing MASs in software infrastructures.

4.1. Self-Organization in Grid and Cloud Computing

Although the idea of grid computing was invented as early as in the 1970s, it was actually put to good use only in the last ten years when computer resources became a commodity required by users for which they are ready to pay. Since it is a commodity, users do not need to know how it operates and where it is located. The same also concerns other resources, such as services, data, and others. From the user point of view, computers should be efficiently utilized, the data must be at the place where they are needed, and services must be easily available and easy to find. The last reason stimulated the appearance of the cloud computing paradigm. Under this paradigm, grid computers and resources of a cloud computing system (memory, services, applications, access interfaces) have a state depending on time when it is not known which of them are available at the current time.

The biological stigmergy, molding, and brood sorting mechanisms discussed above offer possibilities for managing such resources. First, we briefly review the informal ideas and analogies that attract researchers to such self-organization mechanisms, and then consider some most interesting implementations of these mechanisms in agent-based applications in more detail. When searching for services, it is reasonable to assume that the required services that were already used earlier can be also available currently. Then, when a service is requested, it is reasonable to use the search history. This assumption suggests the idea of using swarm intelligence because the task of finding services with account for history is akin to the idea of memory with a positive and negative feedback implemented using pheromones.

There are successful applications of the swarm intelligence mechanism to building self-organizing routing protocols in mobile networks and in P2P networks. A very interesting multiagent-based implementation of such a protocol, namely, the protocol *AntHocNet*, is thoroughly described below in this section.

Molding and brood sorting suggest a good idea for the self-organization of various resources based on the concept of "similarity" (see [17]). For example, sorting makes it possible to organize similar or shared data in the same virtual domains of the cloud computing system. An example of the practical application of such a mechanism is given by the distributed data storage Freenet in which the storage of data and access to them are implemented as a self-organizing overlay (virtual) P2P network. Below, this application is described in detail. Molding in combination with sorting can be used as a self-organization mechanism for solving the resource clusterization problem, for example, for the aggregation of data stored in a certain

place of the cloud computing system. If a data cluster becomes too large, the time needed for searching in it increases. In this case, molding can initiate a new differentiation phase that yields another partitioning of the data set and another data layout.

Cache memory is a part of the Web software infrastructure that makes it possible to minimize the query execution time. From the user side, the cache memory works as a set of proxy servers;[3] it receives requests for information. The cache memory stores copies of requests and the results. Since this memory resides on the web server, a repeated request for the same information can be satisfied from the cache memory, which reduces the response time and the workload on the server. The server cache is used to minimize the workload on its dynamic pages by storing the search results that can be needed later. This protects the server from extra switches during processing, which also reduces the server workload and, as a consequence, reduces the time needed to find other dynamic fragments of data. The cache memory tries to optimize the hit rating, which is calculated as the ratio of the data fragments found in the cache and those fragments that required the search on the web. Due to the limited cache size, some data must be periodically deleted from it, which presents a problem because it is important to leave the data that are likely to be needed again. Although there are many methods for writing data to the cache and removing them, the most difficult problem is to choose a strategy depending on the properties of the flow of requests. Self-organization mechanisms can be used for that purpose. In particular, the swarm intelligence mechanism used by ants for seeking food can be employed for finding such a strategy. To implement this mechanism, one can use a virtual ant as an agent that feeds on repeated retrieval of a specific record from the cache. As long as the virtual ant gets enough food, it remains in the cache (as long as the record is needed, the ant is alive). The ants (records) that starve to death are removed from the cache. To realize adaptation, the ant agents must diffuse a pheromone that could indicate where the resources are likely to be found. Periodically, a new cloud of ant agents is created that follows this pheromone trail in order to predict which resources will be needed in future and to form new records in the cache based on this prediction.

Below, we consider specific applications of self-organizing MASs in grids and in software infrastructures of computer networks.

4.2. Load Balancing in the Grid[4]

Consider the computer network in which each node corresponds to an individual computation resource (computer). Tasks to be served arrive in the grid, and the key job of the central grid management system is to distribute those tasks among the grid nodes so as to best utilize the grid resources (for example, balance the workload).

This problem was studied in [36]. The model of the environment includes flows consisting of independently distributed tasks with Poisson distributions with different parameters; the tasks arrive directly at the grid nodes. These tasks are assumed to have different complexity, but the grid control system does not know their specific characteristics. Furthermore, it is assumed that the computers have different performance. For each node, a stochastic model of its presence in the grid is specified that generates the time instants when this computer appears in the grid and leaves it. The grid network has an ad hoc architecture; that is, each node is aware only about the existence of its neighbors, but it knows nothing about the other nodes; thus, every node can directly interact only with its neighbors. In other words, this is a P2P network. Some neighbors of any node may sometimes be out of the network, but each node knows which neighbors are present at the current time. This information is obtained using a simple message exchange protocol. Other sources of changes in the network topology (e.g., node mobility) are not taken into account in the model under examination because all the related extra features and problems are similar to those arising due to those induced by the variable presence of the nodes in the grid.

For simplicity, we assume that the performance of each grid node is specified in relative units with respect to the performance of a reference computer (node). The computational cost of tasks is specified in a similar fashion: for each task, an estimate of the time needed to execute it on the reference computer is specified. However, it is important that the node does not know the complexity of the tasks executed on it. This complexity is needed only for the simulation system, which calculates the completion time of the execution of the task on a particular computer and determines the time when the execution of the next task can be started. In addition, each task is characterized by the time when it arrives from the environment.

At the current time, every grid node is executing a task, and certain tasks are waiting for execution in its queue. Since different tasks require different resources and computers have different performances,

[3] A proxy server is a server that acts as an intermediary (in this case, between the user and web resources).

[4] This example is a simplified model of balancing workload on grid nodes. Here, it is mainly considered for illustrative purposes.

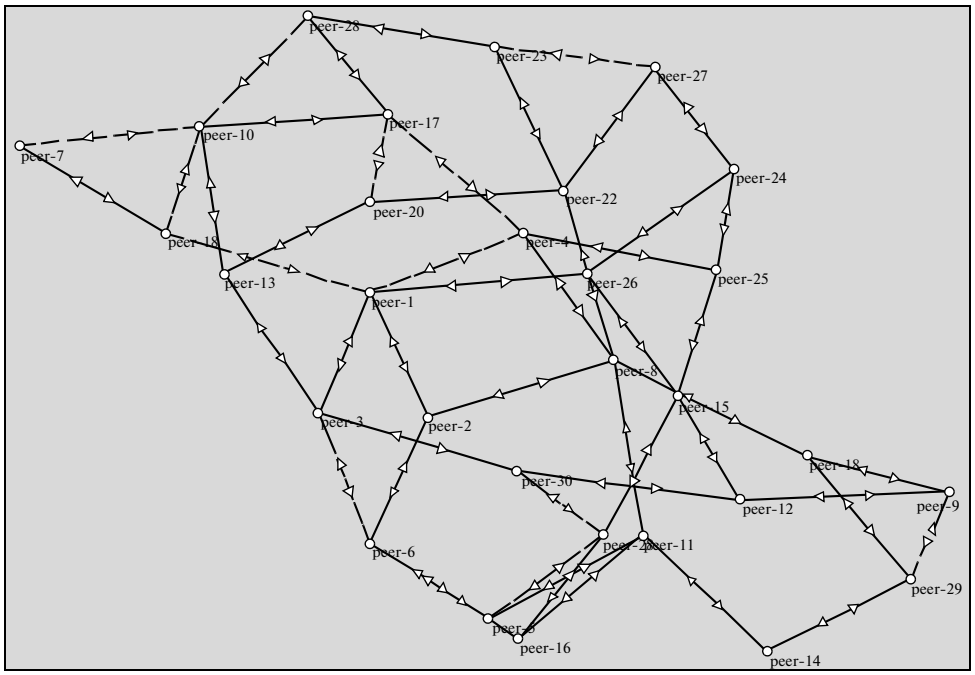

Fig. 3. Self-organizing system for balancing workload of computers in a grid.

some computers in the grid can have a large queue while others can be idle if the grid is not controlled properly. An example of topology of a grid consisting of 30 nodes at a certain time $t=T_k$ is shown in Fig. 3 (see [36]).

The model of the grid load control system is built as a self-organizing MAS. In this model, each grid node is assigned an *agent* that "knows" the length of the current task queue in its computer and can exchange messages with its neighbors, that is, with the agents of the grid nodes that are directly connected with it. The self-organization strategy must initiate the exchange of tasks that could efficiently solve the following two problems:

from the viewpoint of the grid owners, to ensure the most efficient use of the grid resources;

from the viewpoint of users, to ensure the minimal mean service latency time.

The self-organization strategy that ensures a balance of both requirements must have different dynamics for different nodes; otherwise, the mean workload of different computers in the grid will not be balanced. In [36], the self-organization mechanism, the multiagent P2P architecture of the system, the software prototype implementing the dynamic grid resource control, and the results of extensive experiments are described in detail. In the proposed self-organization mechanism, it is assumed that the grid nodes can send tasks from their queues to their neighbors in order to balance the workload. A task that first arrived at a certain computer can be executed or find itself in the queue of any other available computer in the grid. Thus, the executive mechanism of load balancing is the exchange of tasks between grid computers, and the adopted task exchange strategy between the neighboring computers provides the control mechanism.

The essence of the self-organization mechanism is as follows. The exchange of tasks is organized so that the number of tasks sent in one step by each agent to its neighbors is proportional to the number of tasks in its queue and proportional to the number of its neighbors. It turns out that this simple mechanism is fairly effective, which is confirmed by the results of the experiments described in [36]. This mechanism performs well even when the entire input load is concentrated at a single node of the grid and the nodes can appear in the grid and leave it at any time.

The idea of this mechanism is taken from [19], where an abstract network is described in which every node has a number of abstract objects in its possession, and the nodes can exchange these objects. The purpose of the exchange is to equalize the number of objects at all nodes. The exchange of objects is organized so that the number of objects exchanged in one step by each agent with its neighbors is proportional to the number of objects in its possession at the current state. Furthermore, it is proportional to the number of the agent's neighbors; otherwise, the number of objects in certain nodes can permanently increase.

A feature of the strategy just described is that the components of the self-organizing system (agents) interact directly. In accordance with the classification described above such systems are called *direct interaction* systems.

4.3. A Self-Organizing Routing Protocol for Mobile Networks

An important component maintaining the operation of grid networks and other networks with the ad hoc architecture is the packet routing system. For such systems, the MAS paradigm is considered to be very promising. Similar packet routing problems also arise in information systems operating in computer networks with a dynamic architecture and in other non-grid networks. Such networks can include mobile devices, and operate based on wireless communication channels. They are commonly called mobile ad hoc networks (MANET). The conventional packet routing methods are inapplicable in this case due to the fact that the dynamic topology of the network requires that the routing tables be constantly modified, which is not done in conventional routing. To take into account the dynamic topology of the network, all the nodes must somehow be informed about changes in the network. Naturally, this requires extra computational resources, especially when the routing tables must be recalculated often.

Similar problems arise in overlay (virtual) networks built on the basis of peer-to-peer interactions (in P2P networks) in which each node can directly communicate only with a limited number of *neighbor* nodes (at the application level), while it has no information about the other nodes. Often, such networks also have a dynamic composition and topology. In large networks, which is typically the case in real life, the complexity of the required routing algorithms can become insurmountable. This stimulates the search for new principles of routing control.

By now, several routing protocols based on the idea of self-organization in MASs have been proposed (see [37]). The general idea is that the so-called proactive ant packets are sent in anticipatory manner so as to constantly build a current selection of possible routes and update the routing tables at the network nodes in real time. Ant packets move through the network and leave information on their way similar to that contained in pheromone trails (see Section 3). The probability that other packets select a particular route is proportional to the concentration of the digital pheromone on it. This method implements distributed exploration of the environment based on the pheromone accumulation–evaporation paradigm, which leads to the dynamic emergence of structures that form routing tables. Recall that this algorithm actually finds the shortest path, and this is exactly the problem to be solved by a routing protocol.

Among the available protocols of this kind are *ABC* [38], *AntNet* for MANET [39], and the more recent protocol *AntHocNet* [40]. All of them ensure high robustness with respect to errors in individual nodes, ability to balance the load in communication channels, and good adaptation to the dynamics of the network topology and the composition of its nodes. Below, we briefly describe *AntHocNet*, which is the most recent among the three protocols mentioned above and the most efficient in terms of some performance indicators.

The protocol *AntHocNet* explores the network dynamics using the information collected while performing current routing tasks and also uses proactive packets. For these purposes, an agent A is associated with each node; this agent is responsible for the dynamic updating of the routing table. The agent A maintains the list of destination nodes $\{A_{i_1}, \ldots, A_{i_m}\}$ to which it has already sent messages. Some of the nodes in this list are its neighbors. These are the nodes that are directly reachable in the wireless communication environment. For each node A_{i_s}, $s \in 1, \ldots, m$ known to the agent A, a vector of real values F_{i_s} is specified; the number of components of this vector equals the number of the node's neighbors. These components specify the intensity of the digital pheromone on the path to each neighbor under the condition that the message is sent to the node A_{i_s}, which is not necessarily a neighbor. In other words, if a message must be sent from the node A to the node A_{i_s} known to it, it is sent via one of the neighbors, which is chosen stochastically. This mechanism realizes the probability distribution on the set of neighbors in which the components are proportional to the values of the corresponding components of the vector F_{i_s}. Thus, the routing table at the node A is the table of size $m \times n$ consisting of the values of the digital pheromone, where m is the number of nodes known to A and n is the number of its neighbors. The pheromone value in the cell $\langle i, j \rangle$ of the routing table gives an estimate of goodness of the path over the neighbor A_j.

Since the network topology may change with time, this estimate must also be updated dynamically in the course of the network operation. This is done using the so-called *reactive* and *proactive* ant packets. Let us discuss how these packets are used.

If there is a situation in which the node A must send a packet to the node B but the latter node is not registered in the local routing table of the former node, then a *reactive ant packet* is sent first. This is done

in order to find an initial path to B and register it in the routing table. This packet is sent in the broadcast mode using the so-called *flooding protocol*. In other words, the agent of the node that received such a packet analyzes its routing table. If this table contains the destination node B, then the packet is sent via a neighbor chosen stochastically based on this node's probability distribution (of the digital pheromone) for the destination node. If B is not known to the neighbor, it broadcasts the packet to its neighbors as the node A does. This procedure is repeated until the reactive ant packet reaches the destination node B.

When the reactive ant packet reaches the destination, it is converted into a backward ant packet, which travels back to A using the same path on which it was delivered. On its route, it updates the values of the digital pheromone in the routing tables of the nodes on its route. The updating procedure uses a linear combination of the length of the packet queue in each node (for the point in time when this node is passed on the way back) and an estimate of the time taken to deliver the packet to this node. The values of the digital pheromone are updated smoothly; more precisely, its current value is shifted in the direction of the newly found value with a certain weighting coefficient.

If the backward ant packet cannot be delivered to the source (e.g., when the neighbor of the node A over which the backward ant packet is to be delivered left its reachability zone), the source node A assigns to the corresponding entry of the routing table (the digital pheromone) a value indicating that the destination node cannot be reached on this path. All the buffered packets that keep the history of exploring the path to this destination node using reactive and proactive packets are also deleted. Upon such exploration, data packets can be routed to the destination node. They are sent in one direction (without backward packets) in the same way as the proactive ant packets; that is, the next node on the path is selected with the probability proportional to the value of the digital pheromone in the current routing table. It is clear that the probability to select a suboptimal path is lower due to the stochastic selection of the path in the intermediate nodes.

Upon completing the transmission session, the source node periodically sends proactive ant packets with the frequency of one packet per several (not many) data packets. They are routed using a probabilistic choice of the next path segment similarly to how it is done for reactive forward packets. However, the broadcast sending variant is chosen from time to time with a small probability to explore new paths and detect new nodes and neighbors in the network. On the way back, the proactive ant packets update the values of the digital pheromone in the nodes on their way according to the quality determined as described above. In addition, the network nodes periodically sent the pheromone vectors from their routing tables to their neighbors. This is done independently of the information provided by the ant packets. Periodically, the network nodes inform their neighbors about their presence thus indicating that there are alive.

Experimental studies showed that the protocol just described excels other MANET protocols in many respects; in particular, it exceeds the protocol Ad Hoc On-Demand Distance Vector routing (AODV) [41], which was the basic routing protocol for MANET in 2003. Specifically, AdHocNet is superior in the relative number of delivered packets, delivery time, and the average value of the random distortion; however, it is inferior in terms of overheads (the average number of all packets per one data packet).

5. SELF-ORGANIZATION IN OVERLAY STRUCTURES

Recall that an overlay network is a virtual network whose structure differs from the actual communication network underlying the overlay network in question. In an overlay network, every autonomous entity typically has multiple neighbors with which it can communicate "directly" by name as it seems to it in the framework of the overlay network. To enable an entity to interact with other entities in the overlay network, a distributed search mechanism must be used. We emphasize that the overlay network "knows" nothing about the existence of the communication network and its structure. It also does not know how the direct interaction with neighbors is implemented and how the distributed query execution is performed. Presently, the architecture of building information systems using the overlay network paradigm is considered to be promising; it already has many practical implementations and proved to be effective.

To maintain overlay systems, interaction protocols can be used that are implemented in the framework of a special infrastructure that completely separates the application and communication levels and thus maintains interactions in overlay networks. We consider several examples of such self-organization protocols and of existing infrastructures for maintaining applications implemented in terms of overlay networks.

5.1. T-Man Protocol

In [42], a self-organizing T-man protocol for managing the topology of an overlay network is proposed; more precisely, it is designed for optimizing the topology as applied to solving a specific applied or infrastructure problem in the network. In the statement of the problem, a large-scale network with dynamic

architecture in which each node knows about the existence of only a small number of other nodes is considered. It is assumed that the dynamics of the network and the number of nodes in it are such that it is practically impossible to obtain complete information about all the nodes even if there were means designed specifically for that purpose. Depending on the problem (or problems) solved in this network, it is important to know who knows whom. The answer to this question can be completely independent of the physical structure of the communications. Such a situation is typical for overlay networks. The answer to this question determines the quality of problem solution. This question can also be formulated in a different way: who must know whom to solve a problem in the best way? The search of an answer to this question determines the class of problems called overlay network configuration problems. Exactly this problem is solved using the proposed T-man protocol, which implements a simple robust algorithm that performs well on large-scale problems. The algorithm must be simple, robust, adaptive, and applicable to large-scale problems.

The protocol is based on the use of the so-called *ranking function*; according to this function, each node arranges its neighbors by their utility with respect to a performance index of the current network topology. In [42], a simple ranking function representing the distance in a certain space of attributes determined by the interpretation of the optimality of the network topology is examined. This function is used by a node to update the list of its neighbors by deleting from it the least useful ones and adding the most useful (from the viewpoint of the ranking function) neighbors of neighbors. The proposed protocol is implemented using the *Gossip* protocol (see [43, 44]). The properties of T-man are demonstrated using the example of self-configuration of overlay networks designed for sorting, clustering, and dynamic construction of hash tables. A detailed description of this protocol, examples of its application, and convincing experimental results can be found in [42].

5.2. Freenet: A Self-Organizing P2P Network for Accessing a Distributed Data Storage

Freenet [45] is a self-organizing P2P system for data storage that provides access to the data, viewing and writing files, while preserving the anonymity of the authors and data users. Essentially, Freenet is a distributed data storage in which each node provides certain amount of storage for data files; two operations are allowed: adding new data and their viewing. The deletion of data is not possible explicitly. Data are deleted if they have not been accessed for a long time. This storage has no predefined structure; the structure emerges dynamically and changes with time.

Each data file is identified by a key, which is independent of the place where the file is stored. This key is used to find the file and write it. The keys are generated using 160-bit SHA-1 hash functions. The data at each network node are found using a table that stores the addresses of the neighbor nodes and the list of data keys that, as the node believes, are stored in those neighbor nodes. The table content is updated using a local adaptation algorithm. The nodes know only the keys of the data stored in the neighbor nodes, which is sufficient to find them at users' requests. Each table is an analog of an instance of distributed yellow pages in agent-based P2P systems (see [46]); it supports distributed search for files by their keys. However, such a search can be implemented only using a special mechanism (protocol). For this purpose, Freenet uses a mechanism built similarly to the *gossip* protocol [43, 44] complemented with a local self-organization algorithm—*hill climbing algorithm with backtracking* [47]. It finds the next part of the path of the distributed search for the answer. Upon receiving a request from a user or a neighbor node to find a file by its key, the node first looks for this file in its table; if the file is found, the node satisfies the request. Otherwise, it analyzes its routing table, finds the neighbor node that stores the file with the closest key to the key of the file of interest and sends the request to that node. This procedure is called *hill climbing*. It may happen that this node has already been encountered on the way of searching for the answer to the request. In this case, the request is returned to the preceding node, and it finds in its table another not yet used neighbor using the same principle (e.g., the neighbor that has a file with the lexicographically closest key). If it turns out that all the neighbors return the request because they have already be encountered in the search path, the node reports an error and sends a message to its predecessor on the search path, that is, to the node from which the request was received. In turn, the latter node tries to find another neighbor using the same principle (this is *backtracking*).

To ensure that the search process is finite, each request is assigned the maximum number of search steps (hops-to-live); at each step, this number is decremented by one. If the hops-to-live vanishes, the search is terminated, and an error is reported. If the file is found, it is returned to the request source. The key of this file is cached in the node that initiated the request, and the name of the node in which the file is stored is added to the routing table of the requesting node (to the list of its neighbors). If the request source node later receives a new request for this file, it can be satisfied immediately. If this source receives

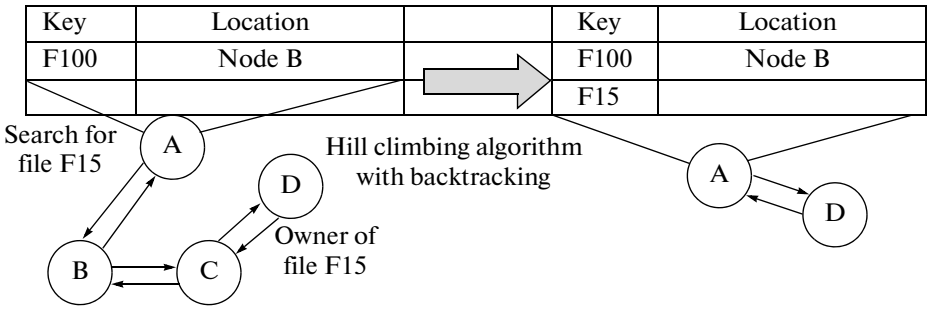

Fig. 4. An illustration of the operation of the self-organizing routing protocol in FreeNet.

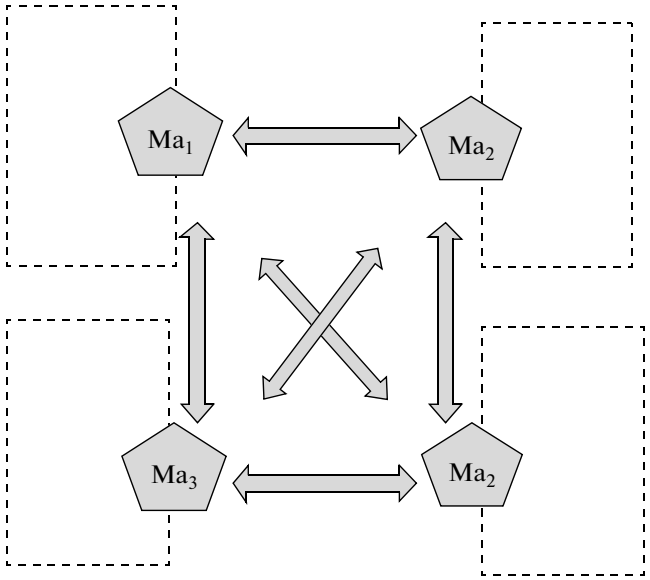

Fig. 5. Configuration of application agents.

a request with a similar key, it will be immediately forwarded to the source of the previously obtained file (see Fig. 4).

This protocol gradually improves the performance of the file search. Since each time a node successfully satisfies a request the attributes of this request (the file key and its location) are added to the node's routing table, the probability that the requests with keys similar to the keys of the requests that were earlier satisfied by this node increases. Furthermore, this node will receive more and more requests with similar keys; therefore, it will successfully satisfy such requests and will be addressed more often. This is a positive feedback. Therefore, the node storages will specialize in storing clusters of similar keys and node addresses that store the corresponding data.

Thus, the creation of a new entry in the routing table upon a successful request routing is the most important point in the node selection algorithm. It leads to the appearance of an emergent property: clusterization of network connections with the nodes containing the most popular instances of data. However, it may happen that several clusters in the network specialize in storing identical (similar) keys, which can lead to time delays because the search is performed in several clusters containing files with similar keys. Nevertheless, experiments confirm the good quality of the described protocol in terms of search speed even when identical data are stored in several nodes.

5.3. Self-Configuration of an Overlay Network of Application Agents

Consider one more model of self-configuration of the overlay network of application agents $\{A_i\}_{i=1}^n$ located at different nodes of the computer network [7] with the architecture shown in Fig. 5. In this network, the application agents exchange messages in the course of the system operation. These messages can

contain requests for services, granting services, declarations of services, and so on. In each node N_j of the network, there is a mediator agent $Ma_j \in \{Ma_k\}_{k=1}^m$. Each application agent is registers with one or several mediators Ma_k.

If an application agent needs a service, it sends a request to its mediator located on the same computer. Upon receiving a request, the mediator tries to find the requested service at the application agents operating on the same computer. If this attempt fails, the search is extended to the agents registered with other mediators. Upon obtaining the search results, the mediator Ma_k analyzes them and adds a mark to the pair service supplier agent–service consumer agent; this mark is positive if the request is satisfied and negative otherwise. For each service, the mediator agent Ma_k "pays" a certain amount of virtual money to the owner of the agent that satisfied the request.

When the number of positive marks for a pair of agents attains a predefined threshold, the mediator Ma_k asks the service supplier agent in this pair to include the corresponding application agent into its own register and offers a "payment" for this. In turn, the owner keeps a record of positive and negative marks of his agents. The number of such marks for the requested agent is used in deciding whether or not to "let go" this agent and how much to charge for it.[5]

Such an agent exchange mechanism implies a regrouping of agents in the nodes of the network; as a result, groups of agents are formed that use the communication space more efficiently. It is clear that the exchange process is dynamic, and the system does not come to a stationary state. There can be various causes for the dynamic behavior. For example, network nodes or agents may leave the network and new agents and nodes can appear; the interests of agents may change, which may require other services or an agent may lose interest in owing certain application agents whose services ceased to be in demand. It is seen that the mediator agents in this network communicate directly and self-organization is based on a market mechanism.

Experimental studies showed that this model performs well in the dynamic situation even when the network is very large. It is especially effective in ad hoc networks and in P2P networks in which the exchange of messages is performed based on peer-to-peer communications.

5.4. Self-Organization in Overlay networks Supported by a Distributed P2P Agent Platform

The development of a functional P2P agent platform (P2P AP) and the related middleware was started by the Nomadic Agent Working Group FIPA at the end of 2005. In its first document [48], the working group declared that no generic P2P AP exists anywhere in the world, and FIPA will make a breakthrough by providing the first complete specifications and by fostering implementations. The efforts of this working group were focused on developing standards for interface specifications for P2P agents installed on mobile devices. The first document describing the architecture of the abstract P2P AP and its interfaces with applications and with the communication infrastructure was released at the beginning of 2006. The first software implementation of this abstract architecture appeared in 2007 (see [46, 49]). Its further development aimed at the use in the ubiquitous computing and ubiquitous communication environment is described below in this section.

The main mandatory functions of the P2P AP provide to all the agents in the network the possibility of semantic search for services and for agents owing those services. Another function is the support of transparent communication between the agents and the service owners.

From the architectural viewpoint, the P2P AP is a distributed set of identical platform instances of which each is installed at a node of the P2P network. Taking into account the basic function of the platform—search for services and for agents owing those services—each instance installed at a node must contain metaknowledge about the agents installed at "its node" and about the services provided by these agents. Therefore, the set of platform instances installed at the nodes of the P2P network is, in essence, a distributed *metaknowledge base* about the agents and their services. The architecture of a developed implementation of the P2P AP instance basically corresponds to the abstract architecture proposed by FIPA in [48] that implements all its mandatory functions is represented in Fig. 6.

The software implementing a platform instance on which the application agents are installed and the component of the node providing the communication service has a three-level architecture (see Fig. 6). The *lower level* contains the software component of the P2P network node called P2P provider. In the current implementation, the P2P provider affords its clients P2P communication channels with other network nodes and implements a mechanism for managing the list of its neighbors in the network. This list is

[5] The metaagent possessing the requested agent can make its decision using an auction mechanism. The details of this mechanism are irrelevant to the present study.

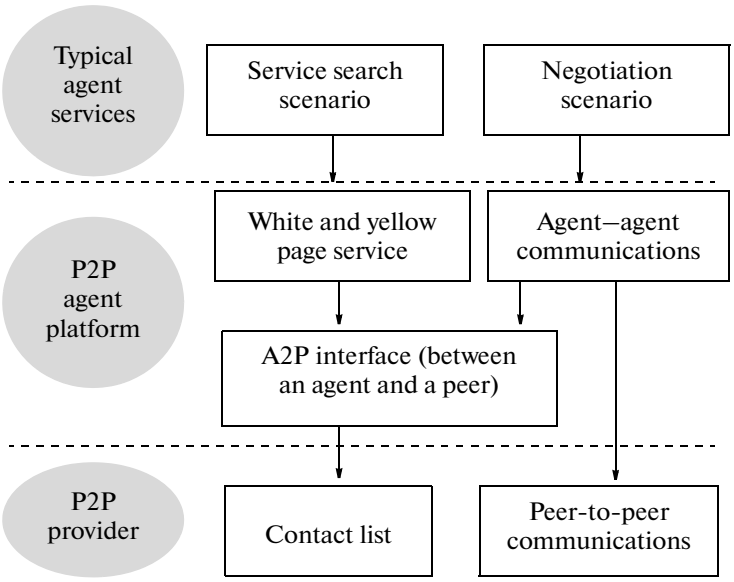

Fig. 6. Functional architecture of the P2P agent platform and P2P provider (peer).

called the *contact list* of the P2P provider. Recall that the contact list contains the names and addresses of the neighbor nodes, that is, the list of nodes in the P2P network with which the node can communicate directly.

The *middle level* corresponds to the instance of the P2P AP that provides the agents registered on this platform with *white and yellow page services.* Each of these services contains a table and a search mechanism. The table of the yellow page service contains information about the names and types of services that are available via this node, that is, about the services that can be provided by the agents installed on top of this platform instance. The table of the white page service contains records with the names of the agents installed in this node and the list of services provided by those agents. The search mechanisms implemented by the white and yellow page services support the search for agents in the P2P network and services, respectively. The search is performed as a distributed procedure that uses the *Gossiop* protocol (see [43, 44]), that is, the same protocol that is used in P2P networks for message routing.

The *upper software level* includes the agents of the application system registered with the P2P AP.

In the same way as the network of P2P providers forms an overlay network installed, for example, on top of the TCP/IP network, the set of P2P AP instances forms an overlay network installed on top of the network of P2P providers (see Fig. 7). An example of practical use of this platform for the software implementation of a self-organizing MAS for load balancing in a grid was discussed in Subsection 4.2.

It is worth noting that the operation quality of each overlay network listed above is characterized by specific performance indexes. At the level of P2P provider network, the quality of solving the routing problem is most important; in this problem, the message sent from the agent platform contains the name of the destination instance of agent platform. Suppose that we deal with a mobile network in which the notion of the node neighborhood in the communication network is determined by the reachability distance of the corresponding communication channel and the structure of the P2P provider network is formed automatically. Optimization in such a network can consist only in the optimization of routing protocol. An example of a self-organizing routing protocol was discussed in Subsection 4.3.

The operation quality of the overlay network of agent platform instances is characterized by the request response time, which directly depends on the structure of this network. It is clear that longer search also results in overloading the communication channels because the duration of search depends on the number of the agent platform instances (nodes of the overlay network consisting of instances of the distributed agent platform) involved in the distributed search. We see strong analogy of this problem with the problem solved in the self-organizing data storage FreeNet (see Subsection 5.2). As in that system, each instance of the agent platform stores data about the services provided by the application agents. Therefore, the same self-organization mechanism that is implemented in the distributed data storage FreeNet can be used to self-organize the overlay system consisting of instances of the agent platform. Another variant is

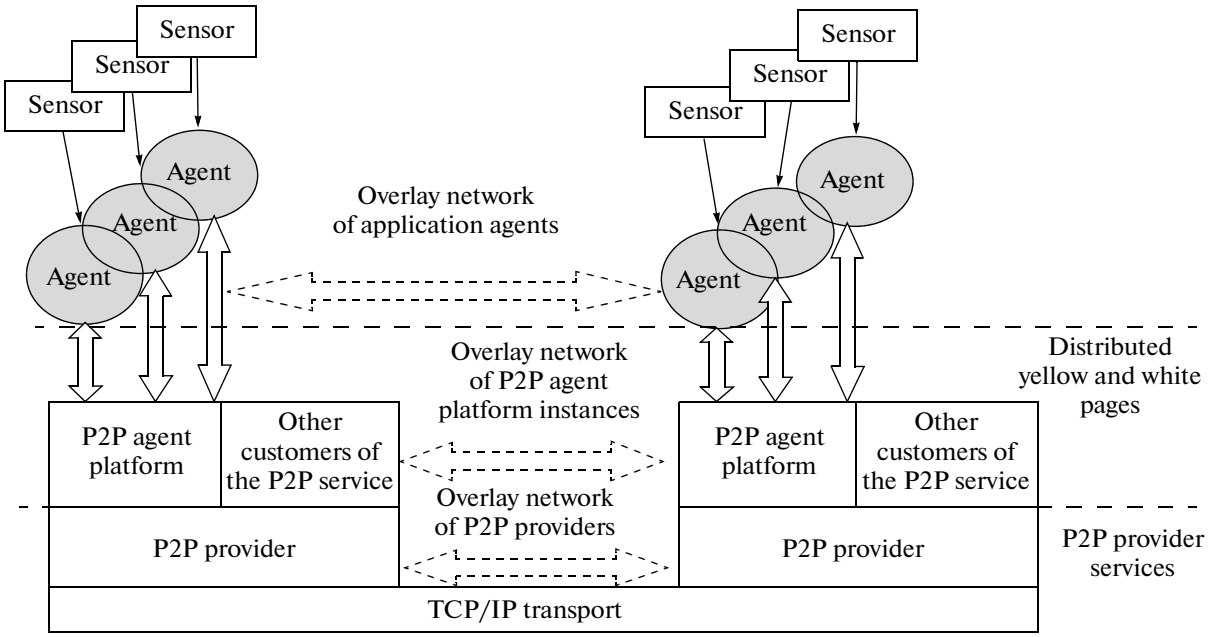

Fig. 7. Functional architecture of overlay networks of P2P providers, instances of the agent platform, and application agents.

the market self-organization mechanism described in Subsection 5.3. Pay attention to the fact that the self-configuration of the network of P2P AP instances is necessary because the composition of the services may change due to the appearance of new services and disappearance of some services from the network. Another cause of the need for self-configuration is the dynamics of the use of different services—the need for specific services can grow or disappear with time.

A similar self-configuration problem arises at the level of application agents. Indeed, the structure of the virtual neighborhood in this overlay network considerably affects the time delays in the operation of the agent platform instances. For example, a request for a service owned by a neighbor in the overlay network of application agents is found very quickly, which reduces the overheads for the search of services and agents. Therefore, it is reasonable to use a self-organization algorithm in this case.

5.5. Ubiquitous Computing

Presently, computing becomes a permanent companion of everyday life. It tends to integrate in a man's environment helping him make decisions, deal with everyday problems, and save time. Such a paradigm of using computers and information technologies is called *ubiquitous computing*. In this paradigm, it is assumed that there is a multitude of unnoticeable computers in the environment surrounding people [50]. Presently, this term is mainly applied to embedded systems, which typically include many distributed computer devices performing simple functions in real time and interacting within the decision making and control system. Typically, these tasks are performed by processors with very limited resources such as speed, memory, and energy consumption. For example, a modern car can include as many as hundreds of processors.

A key problem in this field is the *creation of a self-organizing software infrastructure* for supporting the interaction of heterogeneous devices and computer programs installed in the nodes of the embedded computer network with dynamic topology and the changing set of nodes. Two examples of such an infrastructure were discussed in Subsections 5.2 and 5.3. Additionally, the infrastructure for ubiquitous computing must be adapted to the dynamically changing application *context,* for example, when this application is mobile; this context can be different outside or within a room. Thus, to make decisions, it is important to determine the current context. For the distributed representation and updating of the current context, self-organization based on computational fields was considered in Sections 2 and 3. In [27, 51], specific examples of this kind of self-organization are considered.

Recall that the contextual information in this model is represented by a set of distributed digital computational fields called *co-fields*.[6] Agents can generate various co-fields encoding in them contextual information about themselves and their environment. Other agents can perceive these fields, and decode the meaning of the contextual information stored in them. The agents can perceive the gradients of those fields, for example, by exchanging information with their neighbors, and calculate the direction of motion in the physical space, orient themselves in the virtual space in a certain sense, and the like. This yields a global self-organizing behavior model of the system as a whole. For example, if a visitor to a museum has a hand-held computer or a smartphone, these devices may interact with similar devices of other visitors and with the fields representing the state of various museum resources; these devices can play a double role. On the one hand, they can serve as an individual assistant to the visitor suggesting him where to go and what objects of interest he can find in the proposed direction. On the other hand, collecting the information obtained from the individual computers of the visitors, one can control the global processes of serving the visitors based on the current workload of the museum resources. A similar problem arises in traffic control, search for Web resources, etc. In other words, in many applications operating in a ubiquitous computing environment, the autonomous entities coexisting in this environment can always have information about the current context extracting it based on the co-field model. Examples of using this model for performing ubiquitous computing tasks are discussed in the second part of the present paper.

CONCLUSIONS

The technology of *self-organizing multiagent* systems is a topic of increasing attention in the theory and practice of information technologies. This technology integrates the unique properties of the self-organization model, advantages of the multiagent architecture, and the technologies of its software implementation. The self-organization model is implemented using simple and clear local interactions of the distributed components of the system. It is scalable and can adapt to the dynamics of the external environment, the changing composition and structure of the system itself; it is durable (i.e., it remains operational under multiple failures). Self-organization is one of the principles underlying the operation of living systems. The multiagent technology complements the self-organization model with other unique properties. The main properties of multiagent systems that distinguish them from the software systems based on other architectures are computations based on the interaction of distributed components and the implementation of those interactions in a high-level programming language. We note that these properties determine another (in addition to self-organization) basic principle of living systems operation. Multiagent self-organizing systems integrate both principles, and it seems that this is the source of their great potentials.

The technology of self-organizing MASs is presently considered as the only information technology able to meet the challenges of modern key applications and suggest adequate methods, software architectures, and tools for the development and software implementation of most complex of those applications. It has great potentials in regard to the large-scale applications characterized by openness, autonomy of their subsystems, network organization, and mobility. Such problems cannot presently be solved using conventional approaches and architectures.

This paper describes the state of modern studies and designs in the field of self-organizing MASs based on the literature and the author's studies. It focuses on the principles and mechanisms of self-organization and is based on the expertise of their usage in many practical projects in the organization of computational processes.

By now, many different models and software implementations of self-organizing MASs have been proposed. As of now, the majority of them are at the research stage; however, industrial-level applications start to appear. In this paper, various applications of self-organizing MASs in the field of the intelligent support of various aspects of computer network operation were described. Among them are such applications as network resource management, support of adaptive routing in networks with a dynamic structure and composition (in particular, in networks including mobile devices), support of the ubiquitous computing concept and self-configuration of virtual (overlay) networks, and some others. These applications, which form software infrastructures for maintaining modern distributed (e.g., mobile applications) are most complicated and, at the same time, most needed components of important modern applications. The examples of their research software implementations confirm the great potential of self-organizing MASs. However, they do not exhaust the field of application of multiagent self-organizing systems. This will be demonstrated in the second part of this paper.

[6] By implication, this term should be interpreted as shared fields.

ACKNOWLEDGMENTS

The results presented in Section 5.4 were obtained within program No. 14 of the Presidium of the Russian Academy of Sciences, project no. 214.

REFERENCES

1. Interdisciplinary Program on Application Software towards Exascale Computing for Global Scale Issues // http:www.dfg.de/g8-initiative. Accessed November 11, 2010.
2. W. R. Ashby, "Principles of the Self-Organizing Dynamic System," J. General Psychology, No. 37, 125–128 (1947).
3. P. P. Grassé, "La reconstruction du nid et les coordinations interindividuelles chez Bellicositermes natalensis et Cubitermes sp., la théorie de la stigmergie: Essais d'interpretation du comportement des termites constructeurs," J. Insectes Sociaux **6**, 41–84 (1959).
4. *Self-Organizing Systems*, Ed. by T. N. Sokolov (Mir, Moscow, 1964) [in Russian].
5. G. Nicolis and I. Prigogine, *Self-Organization in Nonequilibrium Systems: From Dissipative Structures to Order through Fluctuations* (Wiley, New York, 1977; Mir, Moscow, 1979).
6. A. B. Kazanskii, "Formalization of Being-with: Bootstrap Systems in Physics, Earth System Science, Cybernetics, and Biology." http://akazansky.by.ru/Kazansky_alm_2007tab.htm. Accessed November 11, 2010.
7. C. Bernon, V. Chevrier, V. Hilaire, et al., "Applications of Self-Organising Multi-Agent Systems: An Initial Framework for Comparison," Informatica, No. 30, 73–82 (2006).
8. C. Bernon, V. Camps, M.-P. Gleizes, et al., "Tools for Self-Organizing Applications Engineering," in *Ser. Lecture Notes in Artificial Intelligence*, Ed. by G. Di Marzo Serugendo (Springer, 2004), Vol. 2977, pp. 283–298.
9. T. De Wolf and T. Holvoet, "Design Patterns for Decentralized Coordination in Self-Organizing Emergent Systems," in *Ser. Lecture Notes in Artificial Intelligence,* Ed. by S. Brueckner, G. Di Marzo Serugendo, A. Karageorgos, et al. (Springer, 2005), Vol. 3464, pp. 28–49.
10. L. Gardelli, M. Viroli, and A. Omicini, "Design Patterns for Self-Organising Systems," in *Ser. Lecture Notes in Computer Science*, Ed. By H.-D. Burkhard, G. Lindemann, R. Verbrugge, et al. (Springer, 2007) Vol. 4696, pp. 123–132.
11. L. Gardelli, M. Viroli, M. Casade, et al., "Designing Self-Organising Environments with Agents and Artifacts: A Simulation-Driven Approach," Int. J. Agent-Oriented Software Eng. **2** (2), 254–271 (2007).
12. L. Gardelli, M. Viroli, M. Casadei, et al., "Designing Self-Organising MAS Environments: The Collective Sort Case," in *Ser. Lecture Notes in Artificial Intelligence: Environments for Multi-Agent Systems III*, Ed. by D. Weyns, H. V. D. Parunak, and F. Michel (Springer, 2006), Vol. 4389, pp. 254–271.
13. J.-P. Georgé, B. Edmonds, and P. Glizes, "Making Self-Organizing Adaptive Multi-Agent Systems Work—Towards the Engineering of Emergent Multi-Agent Systems," in *Methodologies and Software Engineering for Agent Systems* (Kluwer, Boston, 2004), Chap. 8, pp. 319–338.
14. M.-P. Gleizes, V. Camps, J-P. Georgé, et al., "Engineering Systems which Generate Emergent Functionalities," in *Proc. Int. Workshop on Engineering Environment-Mediated Multi-Agent Systems (EEMMAS 2007), Dresden, Germany, 2007*, pp. 58–75.
15. J. Lind, "Patterns in Agent-Oriented Software Engineering," in *Ser. Lecture Notes in Computer Science,* Ed. by F. Giunchiglia, J. Ordell, and G. Weiß (Springer, 2003) Vol. 2585, pp. 47–58 (2003).
16. G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, "Self-Organisation in Multi-Agent Systems," Rapport de recherche IRIT, No. 2005–18 (Universite Paul Sabatier, Toulouse, 2010). http://www.irit.fr/TFGSO/DOCS/TFG2/TFGIISO_LongReport.pdf. Accessed November 11, 2010.
17. M. Mamei, R. Menezes, R. Tolksdorf, et al., "Case Studies for Self-Organization in Computer Science," J. Syst. Architecture **52**, 443–460 (2006).
18. T. De Wolf and T. Holvoet, "Emergence versus Self-Organisation: Different Concepts but Promising when Combined," in *Ser. Lecture Notes in Artificial Intelligence. Engineering Self-Organizing Systems: Methodologies and Applications*, Ed. by S. Brueckner and G. Di Marzo Serugendo, A. Karageorgos, et al. (Springer, 2005), Vol. 3464, pp. 1–15.
19. A. Omicini and L. Gardelli, *Self-Organisation and MAS: An Introduction*. http://unibo.lgardelli.com/teaching/2007-selforg-mas.pdf. Accessed November 11, 2010.
20. G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, "Self-Organization in Multi-Agent Systems," J. Knowledge Eng. Review **20** (2), 165–189 (2005).
21. G. Di Marzo Serugendo, M.-P. Gleizes, and A. Karageorgos, "Self-Organisation and Emergence in Multi-Agent Systems: An Overview," Informatica **30** (1), 45–54 (2006).
22. M. Mamei, M. Vasirani, and F. Zambonelli, "Self-Organizing Spatial Shapes in Mobile Particles: The Tota Approach," in *Proc. Int. Conf. on Engineering Self-Organizing Systems, Methodologies and Applications (ESOA 04), New York, 2004*, pp. 138–153.

23. J.-P. Mano, C. Bourjot, G. Lopardo, et al., "Bio-Inspired Mechanisms for Artificial Self-Organised Systems," Informatica **30** (1), 55–62 (2006).

24. K. Tumer and D. Wolpert, "A Survey of Collectives," in *Collectives and the Design of Complex Systems* (Springer, 2004), pp. 1–42.

25. J.-P. Georgé, M.-P. Gleizes, P. Glize, et al., "Real-Time Simulation for Flood Forecast: An Adaptive Multi-Agent System STAFF," in *Proc. Symp. on Adaptive Agents and Multi-Agent Systems, University of Wales, Aberystwyth, UK, 2003*, pp. 7–11.

26. M. Mamei and F. Zambonelli, "Motion Coordination in the Quake 3 Area Environment: A Field-Based Approach," in *Ser. Lecture Notes in Computer Science*, Ed. by D. Weyns, H. V. D. Parunak, and F. Michel (Springer, 2005) Vol. 3374, pp. 264–278.

27. M. Mamei, F. Zambonelli, and L. Leonardi, "Co-Fields: A Physically Inspired Approach to Motion Coordination," Int. J. IEEE Pervasive Computing **3** (2), 52–61 (2004).

28. A. Chavez and P. Maes, "Kasbah: An Agent Marketplace for Buying and Selling Goods," in *Proc. First Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, 1996*, pp. 75–90.

29. K. Fisher, J. P. Mueller, I. Heimig, et al., "Intelligent Agents in Virtual Enterprises," in *Proc. First Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, 1996*, pp. 205–224.

30. N. Jennings, P. Paratin, and M. Jonson, "Using Intelligent Agents to Manage Business Processes," in *Proc. First Int. Conf. on the Practical Application of Intelligent Agents and Multi-Agent Technology, London, 1996*, pp. 345–376.

31. T. Sandholm, "Negotiation among Self-Interested Computationally Limited Agents," Ph.D. Thesis (University of Massachusetts, Amherst, 1996). http://portal.acm.org/citation.cfm&id=924453, http://www.cs.cmu.edu/~sandholm/dissertation.ps. Accessed November 11, 2010.

32. G. Smith, "Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver," IEEE Trans. on Computers **29**, 1104–1113 (1980).

33. R. Smith and R. Davis, "Framework for Cooperation in Distributed Problem Solving," IEEE Trans. Syst., Man, Cybernetics **11**, 61–70 (1981).

34. P. Valckenaers, H. Van Brussel, and T. Holvoet, "Fundamentals of Holonic Systems and Their Implications for Self-Adaptive and Self-Organizing Systems," in *Proc. 2nd IEEE Int. Conf. on Self-Adaptive and Self-Organizing Systems, Italy, 2008* (IEEE Computer Press, 2008), pp. 168–173.

35. R. Nagpal, "A Catalog of Biologically-Inspired Primitives for Engineering Self-Organization," in *Ser. Lecture Notes in Artificial Intelligence*, Ed. by G. Di Marzo Serugendo, A. Karageorgos, et al. (Springer, 2004) Vol. 2977, pp. 53–62.

36. V. I. Gorodetskii and O. L. Bukhvalov, "Managing Workload in a Grid Based on the Multi-Agent Self-Organization Model: Part 1. Multi-Agent Model and the Self-Organization Mechanism," Mekhatronika, Avtomatizatsiya, Upravlenie, No. 3, 40–46 (2011).

37. C. Bernon, M.-P. Gleizes, S. Peyruqueou, et al., "ADELFE: A Methodology for Adaptive Multi-Agent Systems Engineering," in *Ser. Lecture Notes in Computer Science,* Ed. by P. Petta, R. Tolksdorf, and F. Zambonelli, (Springer, 2002) Vol. 2577, pp. 156–169.

38. R. Schoon der Woerd, O. E. Holland, J. L. Bruten, et al., "Ant-Based Load Balancing in Telecommunications Networks," Int. J. Adaptive Behavior **5** (2), 169–207 (1996).

39. G. Di Caro and M. Dorigo, "AntNet: Distributed Stigmergetic Control for Communications Networks," J. Artif. Intell. Research, No. 9, 317–365 (1998).

40. G. Di Caro, F. Ducatelle, and L. Gambardella, "AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile ad hoc Networks," Europ. Trans. Telecommun., Special Issue on Self-Organization in Mobile Networking **16**, 443–455 (2005).

41. C. E. Perkins and E. M. Royer, "Ad-Hoc on-Demand Distance Vector Routing," in *Proc. 2nd Workshop on Mobile Computer Systems and Applications, Los Alamitos, Calif.* (IEEE Computer Society Press, 1999) pp. 90–100.

42. M. Jelasity and O. Babaoglu, "T-Man: Gossip-Based Overlay Topology Management," in *Ser. Lecture Notes in Artificial Intelligence*, Ed. by S. Brueckner, G. Di Marzo Serugendo, D. Hales, et al. (Springer, 2006), Vol. 3910, pp. 1–15.

43. M. Kwiatkowska, G. Norman, and D. Parker, "Analysis of a Gossip Protocol in PRISM," ACM SIGMETRICS, Performance Evaluation Review **36** (3), 17–22 (2008).

44. S. Lin, F. Taïani, and G. S. Blair, "GossipKit: A Framework of Gossip Protocol Family," in *Proc. 5th Workshop on Middleware for Network Eccentric and Mobile Applications, Magdeburg, 2007*, pp. 26–30.

45. I. Clarke, O. Sandberg, B. Wiley, et al., "Freenet: A Distributed Anonymous Information Storage and Retrieval System," in *Proc. Int. Workshop on Designing Privacy Enhancing Technologies* (Springer, New York, 2000), pp. 46–66.

46. V. I. Gorodetskii, O. V. Karsaev, V. V. Samoilov, et al., "Development Tools for Open Agent Networks," J. Comput. Syst. Sci. Int. **47**, 429–446 (2008).
47. S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach* (Prentice Hall, Upper Saddle River, N.J., 2003).
48. FIPA Nomadic Agent Working Group (P2PNA WG6). http://www.fipa.org/subgroups/ P2PNA-WG.html. Accessed November 11, 2010.
49. V. Gorodetskiy, O. Karsaev, V. Samoilov, et al., "P2P Agent Platform: Implementation and Testing," *Ser. Lecture Notes in Artificial Intelligence*, Ed. by S. R. H. Joseph, Z. Despotovich, and G. Moro, (Springer, 2010), Vol. 5319, pp. 41–54.
50. M. Weiser, "The Computer for the 21st Century," Scientific American Special Issue on Communications, Computers, and Networks (1991).
51. M. Mamei, F. Zambonelli, and L. Leonardi, "Distributed Motion Coordination with Co-Fields: A Case Study in Urban Traffic Management," in *Proc. 6th Int. Symp. on Autonomous Decentralized Systems (ISADS'03)* (IEEE Computer Press, Washington, 2003), pp. 63–70.