1-2006

Topologies of agents interactions in knowledge intensive multi-agentsystems for networked information services

Qiuming Zhu
University of Nebraska at Omaha, qzhu@unomaha.edu

Topologies of agents interactions in knowledge intensive multi-agent systems for networked information services

Qiuming Zhu *

Department of Computer Science, University of Nebraska at Omaha, 6001 Dodge St., Omaha, NE 68182, USA

Abstract

Agents in a multi-agent system (mAS) could interact and cooperate in many different ways. The topology of agent interaction determines how the agents control and communicate with each other, what are the control and communication capabilities of each agent and the whole system, and how efficient the control and communications are. In consequence, the topology affects the agents' ability to share knowledge, integrate knowledge, and make efficient use of knowledge in MAS. This paper presents an overview of four major MAS topologic models, assesses their advantages and disadvantages in terms of agent autonomy, adaptation, scalability, and efficiency of cooperation. Some insights into the applicability for each of the topologies to different environment and domain specific applications are explored. A design example of the topological models to an information service management application is attempted to illustrate the practical merits of each topology.

1. Introduction

Software agents, one of the most exciting new developments in computer software technology, can be used to quickly and easily build integrated enterprise systems. The software agents, like people, can possess different levels of competence at performing a particular task. The idea of using multiple software agents that communicate and cooperate with each other to solve complicated problems in various complicated personal and enterprise computing application domains on our behalf is intuitively appealing. One significant benefit of multi-agent systems (MASs) is their scalability. Since they are inherently modular, it is easier to add new agents to a multi-agent system than it is to add new capabilities to a monolithic system.

Agents in a MAS can have different functionalities and behaviors. For example, agents can be categorized as self-governing agents, brokered agents, monitored agents, mediated agents, etc. Each individual agent can be crafted to be an expert in solving a specific problem or performing a particular task. A collection of software agents that communicate and cooperate with each other is called an agency. An agency may have a manager that closely supervise and arrange the individual agent's tasks, or may not contain that a closely looking supervisor—like a real estate agency, as long as every agent operates in compliance with the agency operating protocol (e.g. following work ethics, paying fees on time). The underlying agent architecture must support sophisticated reasoning, learning, planning, and knowledge representation of the individual agent or the agencies. A general understanding of a MAS is that: (i) each agent has a partial capability to solve a problem, (ii) there is not necessary a global system control, (iii) data and knowledge for solving the problem are decentralized, and (iv) computations carried out among the agent are asynchronous [13].

MAS contain extremely high-level of software abstractions. Programming an agent-based system is primarily a matter of specifying agent behavior. In MAS, the agents need to work collectively so that, as a group, their behavior solves the overall problem without disruption, conflict, and glitches. When a task is assigned, the agents are likely in needs to find the other agents to collaborate with. Such a task is easy if they know exactly which agents to contact and at which location. However, a static distribution of agents is very unlikely to exist. For dynamic multi-agent systems, agents need to know how and where to find the other agents [16]. The dynamic nature of agent distribution motivates this research to look at the topological models of MAS and study how these models facilitate or hurdle the agent collaborations.

Software developers and system designers use high-level abstractions in building complex MAS. To manage the complexity, MAS abstraction must focus on the important

* Tel.: +1 402 554 3685; fax: +1 402 554 3400.
E-mail address: zhuq@unomaha.edu

and essential properties of a problem and hide the incidental components of that problem. An agent interaction topology provides a simple way of managing the complexity because a topology is essentially a high-level abstraction about the interactions of the functional components in a complex system such as the MAS. The topology of agent interaction also helps to define (or facilitates the definitions of) the communication protocol and the interface among the agents of MAS.

It is understood that in a complex system, each agent only needs to interact with a limited number of agents, most likely the agents in its vicinity. Agents in MAS can be organized and controlled in many different ways. For example, agents could be entitled as equal right citizens. That is, every agent has the same status and control and access right to other agents and their shared resources. In this case, each agent would have the same capability of solving a given problem [3]. Who does what purely depends on who is available at the moment. The benefit of this model is that the system is highly fault tolerant—leave one or two agents out of the cycle, the job still gets done as usual. Moreover, the agents in this model have the maximum degree of autonomy. They volunteer their service by themselves upon a request of service or inbound object/situation/environment changes. One other choice is a hierarchical model in which agents are grouped/labeled with different classes/status in terms of the functionality or assigned rights [28]. These agents are often under a centralized or an upper level control. Some supervisory agent in the system may be identified. This organizational model has the advantage of operational efficiency and configuration flexibility [Sohata94].

Software agents are suitable for use in a wide variety of applications. However, agents can have different ways of interconnections and interactions. Each of the interaction schemes is appropriate for use in implementing certain kinds of applications. Developers must carefully analyze system requirements to determine if the selected agent interaction scheme is an appropriate implementation mechanism. The study of the structural and cooperative topology is necessary for construction of complex systems involving multiple agents and mechanisms for coordination of independent agents' behaviors toward a common goal. MAS can be considered of containing the following four dimensions [11]: (1) Agent granularity (coarse vs. fine); (2) Heterogeneity of agent knowledge (redundant vs. specialized); (3) Methods of distributing control (benevolent vs. competitive, team vs. hierarchical, static vs. shifting roles); and (4) Communication possibilities (blackboard vs. messages, low-level vs. high-level, content). The MAS designers must consider the capabilities of each individual agent and how multiple agents can work together—the architecture and protocol issues. There are many ways and views in the study of multi-agent system architecture and protocol. In this paper the architecture and protocol issues are explored from the topological point of view.

Development of multi-agent system (MAS) applications is often complicated by the fact that agents operate in a dynamic, uncertain world. Uncertainty may stem from noisy external data, inexact reasoning such as abduction, and actions by individual agents. Uncertainty can be compounded and amplified when propagated through the agent system. Moreover, some agents may become disconnected from the rest of the system by temporary or permanent disability of these agents or their communication channel, resulting in incomplete/inconsistent system states. How should we represent individual agents acting in such an uncertain environment, and more importantly, how can we predict how the MAS as a whole will evolve as the result of uncertain inter-agent interactions?

Properly structured topology plays a critical role to address the above problems in MAS systems. The topology determines how the agents interact with human and with each other, what are the relations among the agents, and how data and knowledge are shared and communicated among the agents [18,20]. The topology would also affect the functionality, capacity, and underlying computation mechanisms of the agent assembly. To date, there have been relatively few implementations of complex agent-based systems. The difficulty of determining what agent system topology to employ partly limited the more spacious spreading of MAS in real world applications. A proper topology leads to desirable collective behavior in large and complex MAS. Therefore, MAS research needs an insight on how different architectural topologies of an agent assembly function differently to the effects toward agent adaptation, control, collaboration, and learning [12, Grefenstett296].

In this paper, we first present an overview of four major MAS topology models. They are (1) a Web-like topology where agents are connected (and communicated) as nodes in a complete graph; (2) a Star-like topology where several agents are connected with, and communicate through, a controller/-mediator; (3) a Grid-like topology where each agent is only connected (and communicated) with its neighboring agents, thought the access to other agents or resource not in the neighborhood could be done through the neighboring agents; and (4) a hierarchical collective agent network (HCAN) topology, that combines some of the features of previous models. We assess the advantages and disadvantages of these models in terms of agent autonomy, adaptation, scalability, and efficiency of cooperation. An example of the application of the fourth model for application in information service is presented.

The paper is organized as the following. Section 2 discusses the four major MAS agent cooperation topologies. Section 3 assesses these four topologies in terms of a set of criteria selected. Section 4 presents an analysis of the fourth topologies with respect to different MAS application domains, and points some insights on the applicability of each topology to certain applications. Section 5 presents an exemplar design of using each of the topologies for an information service system application. Section 6 contains conclusion remarks.

2. Taxonomy

Several research communities have modeled distributed computing by studying communication and coordination mechanisms among autonomous software entities, or agents. Agent-based computing focuses on the interaction mechanisms among agents, which permit a rich set of coordinated activities. Effective models of interaction require the following basic capabilities:

(1) A transport mechanism to convey messages in an asynchronous fashion;
(2) An interaction protocol, defining the available types of communications and their semantics;
(3) A content language providing the base for composition of requests and their interpretation; and
(4) An agreed-upon set of shared vocabulary and meaning of concepts (often called on *ontology*).

The degree to which different agents play out distinct roles is certainly an important issue in MAS. The taxonomy presented in this paper is organized along the most important aspects of agents: degree of heterogeneity and degree of communication for interaction and knowledge sharing. The taxonomy is based on the common understanding that: (1) agents are ubiquitous, (2) agents have designated roles, reside at designated place, perform designated tasks for a designated person/controller, and (3) agents can be acting by their own (once deployed) or agents can be acting under coordination of other agents.

The topology of multi-agent cooperation can be classified according to multiple criteria. In this paper, we use the following three criteria to characterize the cooperation:

(1) The ways of activation, supervision, and communication between the agents [18], i.e. how the agents invocate each other, requesting service from each other, and retrieve/pass data to each other;
(2) The dependencies of the agents [19], i.e. whether they function complementary to complete a task, i.e. each functioning on the same course or differently aspects of a course, and
(3) The ways of sharing data, knowledge and other resources, including considerations of at what level they share the data and knowledge to complete a given task [30].

In this section we study four basic MAS topological structures: (1) a Web-like topology, (2) a Star-like topology, (3) a Grid-like topology, and (4) a Hierarchical Collective Agent Network (HCAN) topology. Note that this study is not about the physical links between the agents. Our concern is on the functional links (and interactions) among the agents enabled either by physical links or by virtual communication channels. The four MAS topologies of our study are described in the following.

2.1. Web-like topology

A Web-like topology is featured with a uniform interconnection of the agents in a cooperative environment. That is, every agent node can have directly interaction with all other agent nodes. Usually, these interactive agent nodes form a complete graph, as shown in Fig. 1.

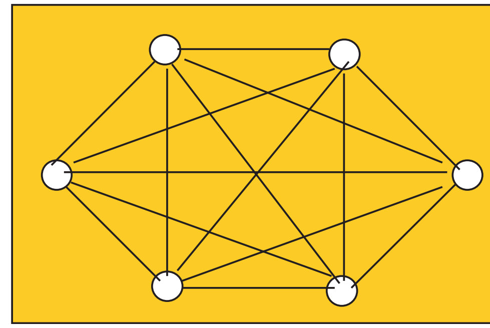

Fig. 1. Web-like topology of agent interaction.

In the Web-like topology, the collection of distributed agents acts as equal members of the community. In this topology, all of the agents have the same internal structure as well as operation goals, domain knowledge, and possible action choices. They also have the same procedure for selecting among their actions. The only differences among agents may be their sensory inputs and the actual actions they take: they may be situated differently in the world or in different environmental settings. Although the agents have identical capabilities and decision procedures, they may have limited information about each other's internal state and sensory inputs. Thus they may not be able to predict each other's actions.

The Web-like topology can also be formed in virtual when the MAS employs an agent-activation scheme called request-and-service protocol, a blackboard kind of communication and task activation approach. In the request-and-service protocol, every agent in the MAS can response to a call issued by one of the agent and perform the task requested, and could be called by other agents to perform specific tasks. That makes the agents seemed all connected directly.

In the Web-like topology, the agents are empowered as equal-right citizens in a MAS society. Every agent receives the same command and request, share the same data and resources, and act at the same level (though functioning differently in terms of the problem to be solved). Each agent can call any other agents, and be called by any other agents. The General Magic's MAS model is a representative example of this kind of topology [34]. General Magic models MAS as an electronic marketplace that lets providers and consumers of goods and services find one another and transact business. This marketplace is modeled as a network of computers supporting a collection of places that offer services to mobile agents. All agents have the same capability to travel from one place to another, to meet other agents which allows them to call one another agent's procedures, to create connections to allow an agent to communicate with another agent in a different place, and to have authority to indicate the real-world individual or organization that the agent represents. Note that in Web-like topologies, agents can perform their service by themselves autonomously upon a request of service (ROS) or inbound objects or situation/environment changes.

A number of variations to the Web-like model exist. For example, the agents are organized in groups (subsets) and agents in each subset are fully connected in the Cougaar MAS architecture. The Cougaar architecture supports a distributed plan, similar to a partitioned blackboard, which is interconnected but not replicated across the agent society [9]. This means that information is shared among only the interested parties. This simple concept, combined with some proven concepts of locality of reference, minimizes the communication requirements and makes possible a managed agent network required of large-scale distributed systems.

2.2. Star-like topology

Unlike in Web-like topology where agents can be cooperative in their own all together by some implicit agreement or activation protocol, there may be actions that *require* explicit coordination for successful execution. In a star-like topology, the activities of the agents are coordinated or administered by some supervisory (or facilitator) agents designated in the assembly. Only agents that have connections built and specified to the coordinator can interact with each other. That is, the agents are more under control and stipulation than those in the Web-like topology. In this topology, functional invocation and data communication is often brokered through connections to one or more facilitating agents. The facilitator is responsible for matching requests from users to agents, with descriptions of the capabilities of the agents in its possession. A structural diagram of such topology is shown in Fig. 2, where the center nodes in dark color denote the coordinators.

Most agent architectures contain specialized agents that are suited for specific operations within the application domain and environment. Often sophisticated systems of application were decomposed into modules, each of which was then transformed into an agent or multi-agents. These agents then are divided into different groups. Agents in each group are capable of performing a specific kind of tasks. In this configuration, the agents may not communicate with each other directly. A supervisor, controller, or mediator is then needed to distribute and coordinate the tasks. Examples of such control agents include (1) the SRI's OAA facilitator [24]; (2) the CMU's RETSINA Matchmaker [32]; and (3) the Infosleuth's Broker [26].

In SRI's Open Agent Architecture (OAA), the facilitators are responsible for matching requests from users and agents,

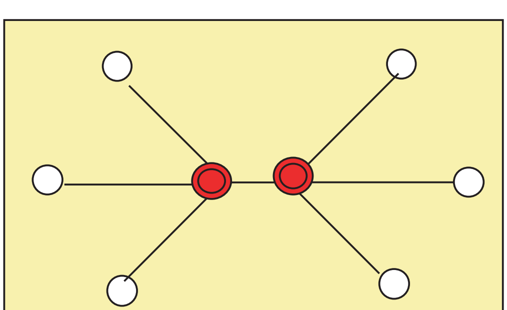

Fig. 2. Star-like topology of agent cooperation.

with descriptions of the capabilities of other agents, and then delegate the tasks to qualified/available agents [8]. Thus, it is not generally required that a requester (user or agent) know the identities, locations, or number of other agents involved in satisfying a request. Facilitators are not viewed as centralized controllers, however, but rather as coordinators, as they draw upon knowledge and advice from several different, potentially distributed, sources to guide their delegation choices. This scheme makes it possible for software services to be provided through the cooperative efforts of distributed collections of autonomous agents.

In a distributed agent framework of Star-like topology, a dynamic community of agents, where multiple agents contribute services to the community, is often conceptualized. When external services or information are required from a given agent, instead of calling a known subroutine or asking a specific agent to perform a task, the agent submits a high-level expression describing the needs and attributes of the request to a specialized facilitator agent. The facilitator agent will make decisions about which agents are available and capable of handling sub-parts of the request, and will manage all agent interactions required to handle the complex query. One advantage of this quasi-distributed agent architecture is that it allows the construction of MAS that are more flexible and adaptable than the fully distributed object frameworks such as those in the Web-like topology. Individual agents can be dynamically added to the community easily, extending the functionality that the agent community can provide as a whole. The agent system of Star-like topology is also able to adapt to available resources in a way that hard-coded distributed objects systems cannot.

One of the important issues to consider when designing a multi-agent system is whether the different agents will be benevolent or competitive. Even if they have different goals, the agents can be benevolent if they are willing to help each other achieve their respective goals [15]. On the other hand, the agents may be selfish and only consider their own goals when acting. In the extreme, the agents may be involved in a zero-sum situation so that they must actively oppose other agents' goals in order to achieve their own. The Star-like topology is more empowered to solve these kinds of goal and action conflicts among the group of agents.

2.3. Grid-like topology

In a grid-like topology, each agent cooperates with a group (an agency) of agents in its neighborhood (in terms of functional connections) that is a subset of agents in the assembly (or community). Each agent has direct connections (in terms of cooperation behavior) to the agents in its neighborhood group (logically, not necessary physically or geographically). Each group may be administered by a supervisor/facilitator designated. Interaction to agents not residing in the neighborhood must pass through the facilitators of the neighborhoods. Such interaction may pass multiple agents in cascade. The designation of facilitator may be changed dynamically in terms of the efficiency of interaction it

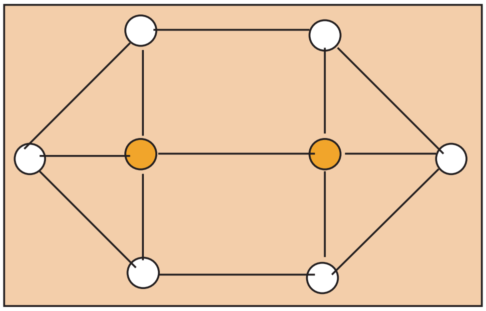

Fig. 3. Grid-like topology of agent cooperation.

enables. Fig. 3 shows a diagrammatic illustration of this topology, where the nodes in dark color denote facilitators under current designation.

Simply described, a grid-like topology is an environment consisting of areas. Areas are required to have exactly one local area coordinator, which is an agent that acts as a facilitator for other agents within its area. Agents can be identified as being inside an area if they have registered with the area's local coordinator. Agents will use the services of local area coordinators to access other agents in the system. Agents can advertise services and find out about other agents' services by means of agent registry or yellow page servers. Agents requiring data sharing with other agents can join virtual environments called cooperation domains, which are supported by cooperation domain server agents.

The agents in Grid-like topology form a more federated agents society. It has relatively low communication and computational requirements, meaning that there are virtually no constraints on the system size. The simplicity of agent interactions also makes it amenable to quantitative mathematical analysis. Each group of agents has a meta-agent that serves as the agent/task manager, which decomposes a task and distributes it to the individual functional agents or other agent managers. Example of MAS in the grid-like topology can be seen at the Object Manager Group (OMG)'s Model [33]. This model is composed of agents (i.e. components) and agencies (i.e. places) as entities that collaborate using general patterns and policies of interaction. Under this model, agents are characterized by their capabilities (e.g. inference, planning, and so on), type of interactions (e.g. synchronous, asynchronous), and mobility (e.g. static, movable with or without state). Agencies, on the other hand, support concurrent agent execution, security and agent mobility, among others.

In many systems, hierarchically organized collections of planning agents that are committed to one particular planning problem are deployed. For example, in MPA- Multi-agent Planning Architecture of SRI [35], the activities of these agents are coordinated by meta-PAs (PAs that control other PAs) with specialized knowledge about strategies for division of labor, conflict resolution, and (in the future) plan merging. Each meta-PA is responsible for coordinating activities among its collection of PAs and other planning clusters.

2.4. HCAN topology

A fourth topology, named a hierarchical collective MAS model, is presented in this section. The hierarchical collective agent network (HCAN) topology of agent cooperation is shown by diagram in Fig. 4. Main properties of the HC topology are (1) Agents are grouped in layers, (2) the layers are organized in hierarchy, (3) agents in each layer are not connected, (4) agents between layers are fully connected, and (5) the control and coordinate of the agent at each layer are through the agents at the higher level.

In the HCAN, agents at the lower level (the data managing module) interface directly to individual sensor/information resources. These agents act in a distributive fashion to process conceptual queries, filter retrieved information using simple proposition logics, and extract useful information as instructed by upper-level (the reasoning or user interface modules) agents. The agents at the upper levels coordinate the activities of the agents at the lower levels using a centralized goal-driven control strategy. They issue conceptual queries, perform data integration and knowledge extraction, and make cross-reference of the information retrieved. The coordinate agents at these levels will apply certain data analysis models and employ reasoning-integration technique to fuse information reported by retrieval agents at the lower levels. Special human-system interfacing agents will provide continual support for interactions between user and the systems, and provide intelligent and dynamic information summarization, annotation, and presentation based on the user-originated inputs and queries.

The major functionalities and design tradeoffs of the HCAN topology are as follows. The HCAN topology is flexible in terms of the ability in which communities of agents can be assembled, and the flexibility with which services can be added at runtime and brought into use without requiring changes to the other part of the agent assembly. A unified set of concepts, declarations and interfaces that are consistent across all services in the framework, and the role played by the agents at different levels are defined. The HCAN topology strikes a balance between the centralized control and distributed computation by allowing distributive agent operations within layers of the hierarchy and enforcing centralized control between the layers of the hierarchy, thus eases the coordination and control needed to manage interactions between agents.

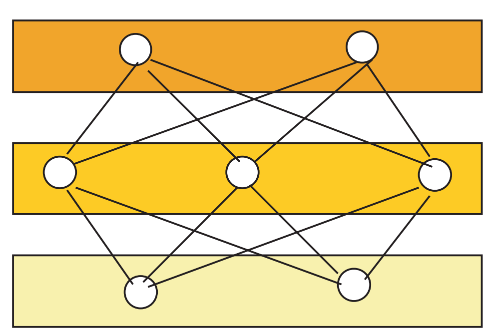

Fig. 4. Hierarchical collective topology of agent cooperation.

Table 1
Features of four major MAS topology

	Web	Star	Grid	HCAN
Center controller /mediator?	No	Yes	Partly	Partial
Agents all at equal level?	Yes	No	No	No
One to all interaction?	Yes	No	No	No
Complete communication link?	Yes	No	Partly	Partial
Local/global distinction?	No	No	Yes	Yes
Automatic service response?	Yes	No	Partly	Partial

The rationale behind the HCAN topology is again the concept of shared and distributed intelligence. It is not a good idea to develop agents with capability of doing everything. Agent must be task-specific for doing something, and for doing some small things really well. That is, agents are specialists on special tasks. For example, it is not necessary to require an agent to possess all the perception, action, and reasoning components, which are necessary for being autonomous and adaptive. Rather, it can be an agent system in which there are agents responsible for perception, agents responsible for action, agent responsible for reasoning, and agents responsible for learning and augment the knowledge of the other agents or accumulate and store the knowledge to a place that are accessible by all the agents. Where the perception agents feed the reasoning agents, the reasoning agents feed the action agents, and the learning agents feed both the reasoning and action agents, etc. Thus, the functionality of an agent must always be limited to a specific domain, on a specific task. That is, based on this observation and understanding the MAS comes into play.

2.5. Summary

Table 1 Summarizes the structure characteristics of the above four MAS topology.

3. Analyses

In this section we explore the advantages and disadvantages of the topologic models of the above in terms of their effects to agent autonomy, adaptation, communication, learning, and efficiency of cooperation. The topology should facilitate the intensive knowledge embedding, accumulation, and incorporation for MAS. A multi-agent system is dynamic in nature, meaning that agents can be added to it or removed from it from time to time. Thus, an agent system topology must also facilitate the dynamic property of agents. The study here focuses on how the specific topology boosts or attenuates the major agent features and functionalities required by MAS, based on a set of agent properties defined as the following:

(1) *Autonomous.* It is known that agents, whether in a MAS or stand-alone, should be proactive, goal directed and act on their own (self-starting behavior) or perform tasks on some user's behalf. Effectiveness of goal achieving is one important property of agents.
(2) *Cooperative.* Agents in a MAS should be specially equipped with the ability to work with other agents to achieve a common goal. They must behave effectively at both self-organizing and delegating states, effective under coordination and negotiation, and conscious of conflict resolution.
(3) *Trustful*. The agents must be reliable when exerting their autonomy in performing the tasks designated by human. They must perform the tasks and complete the tasks in the quality and time as the human instructed.
(4) *Flexible*. Agents in MAS should be flexible in terms of system reconfiguration and task delegation. Agents should be able to join and participate the cooperation community at any time, i.e. dynamic inhabitation. Configuration flexibility leads to scalability that is also critical to MAS operating in dynamic environment.
(5) *Adaptive.* Agents should have a certain level of ability to selectively sense and act/re-act to the environmental situation changes, and should be readily/easily transplantable to different environmental applications.
(6) *Interactive.* Most agents are required to communicate and interoperate efficiently with humans, other systems, and information sources. Agents in MAS must be especially capable of dealing with the complexity issues of resource sharing, distribution, and deadlock breaking.
(7) *Reactive*. The ability to learn and improve the functionality with experience is a very desirable feature of agents. Agents able to dynamically adapt to and learn from the environment will have better capability to adapt to situation/environment changes.

3.1. Web-like topology

Both advantages and disadvantages of the Web-like topology are associated with its indiscriminative behavior of agent activation. The Web-like MAS topology facilitates parallelism and entitles redundancy. While parallelism is achieved by assigning different tasks or abilities to different agents, robustness is a benefit of multi-agent systems that have redundant agents. If control and responsibilities are sufficiently shared among different agents, the system can tolerate failures by one or more of the agents. Domains that must degrade gracefully are in particular need of this feature of MAS: if a single entity -processor or agent- controls everything, then the entire system could crash if there is a single failure.

One question often asked of this kind of MAS is that in such a closely coupled relation among agents—agent network, can agents be really equal members of a society? Or, is this especially good for the joint functionality of a MAS? The answer may depend on what application domain the agent system works in. Although multi-agent systems are often described as being intrinsically more robust than a single agent by virtue of redundancy, fault tolerance is not a natural byproduct of duplication but only emerges through careful design. A complex MAS cannot always be created through cloning a group of single agents designed for the same task.

There has to be some awareness, either on the part of the agents or the system designer, of the role that other members will play in completing the task. Unless the global task is somehow partitioned among the agents, they will either interfere with each other or converge on a sub-optimal division of labor. Thus, the reason why a complete-graph kind of topology is not necessary, and probably undesirable, is that the global interaction with all agents in a domain or application environment is likely not necessary. Moreover, the design of that kind of global interaction system is too complex to deal with. The functional structure of individual agent in Web-like topology is also most complex among the topologies because the agent there needs to know how to communicate with the others, while in other topologies the communication can be handled by the facilitator or broker agent.

3.2. Star-like topology

An advantage of star-like topology is its loosely enforced control and coordination. Though control and coordination limits the boundary of cooperation the agents can reach, it is desirable when efficiency of cooperation is a main issue that needs to be ensured. The star-like topology is suitable for the environment and applications where the MAS is to act as a central planner, that involves team negotiation and needs awareness of what each agent knows and does. It also possesses functional suitability and self-consciousness—each agent is dissimilar in functionality, the dissimilarity determines and distributes tasks. The use of facilitators in OAA offers both advantages and weaknesses with respect to scalability and fault tolerance [6]. For example, on the plus side, the grouping of a facilitator with a collection of client agents provides a natural building block from which to construct larger systems. On the minus side, there is the potential for a facilitator to become a communication bottleneck, or a critical point of failure.

In Star-like topology, the control agent focuses on the interaction mechanisms among agents, which permits a rich set of coordinated activities. Effective models of interaction require some basic capabilities: (1) a transport mechanism to convey messages in an asynchronous fashion, (2) an interaction protocol, defining the available types of communications and their semantics, (3) a content language providing the base for composition of requests and their interpretation, and (4) an agreed-upon set of shared vocabulary and meaning of concepts (often called on *ontology*). Some MAS use game theoretic model for multi-agents cooperation and rely on the assumption that all agents are fully rational. In general, for a set of agents to cooperate, there is a need for a shared ontology among them. It is more critical to have a shared ontology for agents to inter-operate without passing through a facilitator.

Another advantage of mediated topology is that it is easy to define a system in terms of agent-mediated processes. The moderated multi-agent systems are particularly well suited to process and workflow automation, electronic commerce, distributed problem solving, Internet applications.

3.3. Grid-like topology

The grid-like topology makes a tradeoff between increasing the number of agents that can interact directly with each other and retain control of monitoring of agent activities in a reasonable range. The approach is suitable for MAS designed to operate in a well-defined global environment and objectives. The topology entitles the relative merits of model-free and model-based methods. Consider the facilitating of local or networked configuration of the MAS as another criterion, the grid topology is advantages than the other topologies of MAS.

The locally interacted agents in Grid-like topology may demonstrate complex group behavior advantages over the fully connected agent assembly. When agents have similar goals, they can be organized into a team. Each agent then plays a separate role within the team. With such a benevolent team of agents, one must provide some method for assigning different agents to different roles. This assignment might be obvious if the agents are very specific and can each only do one thing. However in some domains, the agents are flexible enough to interchange roles.

3.4. HCAN topology

The HCAN topology makes a tradeoff between distributive and centralized control of multiple gent systems. The collective nature of the agents in the HCAN paradigm overcomes some of these difficulties, for example, relieving the burden of data-exchanges between fellow agents by limiting agent communication to vertical layers of the assembly only. The collective nature of agent relation in the hierarchical architecture simplifies the functional design of the agent interactions and enhances the security and efficiency of the information processing.

Basically, the HCAN is desirable when the MAS is required to have the following functionalities.

(1) A flexible software architecture for accommodating system augmentation and evolutions;
(2) A powerful representation schema for accommodating heterogeneous forms of information;
(3) A diverse interface for various input resources, output formats, and human interactions;
(4) An ability of reasoning on incomplete and inconsistent information, and extracting useful knowledge from the data of heterogeneous resources;
(5) An ability of incorporating real-time dynamics of the information resources into the system anytime during the operation, and promptly adjusting the reasoning mechanisms;
(6) An ability of summarizing and refining knowledge extracted, and distinguishing mission and time critical knowledge from insignificant and redundant ones;
(7) A capability of supplying meaningful and accurate explanations, both qualitatively and quantitatively, of the automated system actions; and

Table 2
Assessment of the topologic models

	Web	Star	Grid	HCAN
Autonomy	5	1	3	4
Cooperative	2	5	3	4
Trustful	1	5	5	5
Flexible	5	5	5	4
Adaptive	2	5	5	5
Interactive	3	1	3	5
Reactive	2	5	3	5

(8) A capability of providing adequate control and scrutinizing of the system operations under the environmental constrains of the given situation.

There is a need for mechanisms for advertising, finding, fusing, using, presenting, managing, and updating agent services and information in most MAS applications. To address these issues, the notion of *middle agents* was proposed [11,22,23]. Middle agents are entities to which other agents advertise their capabilities, and which are neither requesters nor providers from the standpoint of the transaction under consideration. The advantage of middle agents is that they allow MAS to operate robustly when confronted with agent appearance, disappearance, and mobility. There are several types of agents that fall under the definition of middle agents. Note that these types of agents, which are described below, are defined so vaguely that sometimes it is difficult to make a clear differentiation between them.

- ☐ *Facilitators.* Agents to which other agents surrender their autonomy in exchange for the facilitator's services. Facilitators can coordinate agents' activities and can satisfy requests on behalf of their subordinated agents.
- ☐ *Mediators.* Agents that exploit encoded knowledge to create services for a higher level of applications.
- ☐ *Brokers*. Agents that receive requests and perform actions using services from other agents in conjunction with their own resources.
- ☐ *Matchmakers and yellow pages.* Agents that assist service requesters to find service provider agents based on advertised capabilities.
- ☐ *Blackboards*: Repository agents that receive and hold requests for other agents to process.

The HCAN provides a proper balance on the need of centralized and distributed middle agents for the control and coordination of the multi-agents in the complex system.

The assessments of the four major topologies are summarized in Table 2. We give a rating of 1–5 to each of the performance measurements for each topology, where a rating of 1 is the lowest and 5 is the highest. The assignments are somehow subjective.

4. Applications

After comparing the four basic topological structures and their pros and cons, we can now relate the major topologies to the diverse sets of MAS applications. It is noted that most of the agent research and development up to date are in the area of agent modeling and agent building tools. Wide spreading true applications are still lacking. Over hundred agent construction toolkits, development environment, or component libraries can be returned from a simple search on Internet. Chauhan and Baker, 1998's JAFMAS supports directed (point to point) communication as well as subject based broadcast communications [5]. Ciancarini et al [7] introduced PageSpace as a referential architecture for designing interactive multi-agent applications, using variants of the coordination language Linda to guide their interactions. Several kinds of agents live in the PageSpace: user interface agents, personal home agents, agents that implement applications, and agents that interoperate with legacy systems. Suzuki et al. [31] proposed 'self-migrating threads' as a new cluster-computing paradigm for multi-agent applications, which can be viewed as the interactions among autonomous computing entities, each having its own objectives, behavior, and local information in a synthetic world. Self-migrating threads have both navigational autonomy of mobile agents and fine computation granularity of threads. In ZEUS [25], coordination is supported through use of conversation classes that agents utilize to manage their interactions with other agents during problem solving. The conversation classes implement rule based automata models, similar in spirit to the way co ordination behavior is managed in ZEUS.

Multi-agent systems (MASs) provide for the modeling of practical systems in the fields of communications, flexible manufacturing, and air-traffic management [4,27]. Some of the previous work in multi-agent system development concentrated on domain-independent frameworks, standard protocol definitions, some handling of uncertainty and utility, and extensive models of collaboration [16]. However, there lacks methods for solid decision-theoretic model of agents learning, adaptation, control and collaboration. Arai *et al* presented a reinforcement learning approach known as Profit-sharing that allows agents to learn effective behaviors with in dynamic and multi-agent environments [1]. The increased prevalence of agents raises numerous practical considerations. Three of these are (1) adaptability to unforeseen conditions, (2) behavioral assurance, and (3) timeliness of agent responses [2,14]. Two questions are always asked about any type of technology. (1) What advantages does it offer over the alternatives? And (2) In what circumstances is it useful? The same questions apply to the study of topologies of MAS. The evolution of Multi-Agent Systems and the growing interest in multi-agent development platforms have leaded to some interesting tools for agent software developers. Although, some platforms are grounded on well-known models, platforms for development of agents are widely heterogeneous globally. Questions remaining: What topology of agent interaction is good for what kind of applications?

We first take a look at some examples to see the diversity of MAS applications and what kind of cooperation topology is needed for each of the applications.

1. An electronic commerce application might have buyer agents, seller agents, stocking agents, database management agents, email agents, etc. A loan approval application ties together branch banks, the main bank, loan underwriting companies, and credit reporting companies, and automates much of the loan approval process. All of these agents involve distributed computation or communication between components, need to communicate with each other, and must have the capability of working together to achieve a common set of goals. Multi-facets of considerations must be made with respect to the differences in performance efficiency and competency when choose proper topology for the agent system in these applications.
2. Data fusion and mining applications that reason about the messages or objects received over a network require multi-agents organized in sequences of work-flow and coordination, e.g. network interfacing agent, information searching agent, recording agents, inference agents, reporting generation agents, etc. The same situation applied to e-collaboration and e-learning applications. Agent system in these applications must balance the distributiveness and centralized control.
3. Automation applications for example in plant and process automation, workflow management, robotics including Unmanned Autonomous vehicles (UAV), etc. requires the agent to be capable of operating without much user input or intervention. An embedded factory controller might consist of a user interface agent, a database interface agent, a machine tool interface agent, and a process monitoring and control agent. All of these agents could run concurrently on the same processor or could be easily distributed across multiple processors.
4. There are applications that require significant communications between components for sensing or monitoring of the environment, making decisions and performing autonomous operations. Since the agents in these applications need to have the ability to reason (i.e. draw inferences), they can easily perform sequences of complex operations based on messages they receive, their own internal beliefs, and their overall goals and objectives. For example, email and instant messaging system that uses software agents to implement the mail client. The system is designed to ensure that messages remain private. Privacy is assured because messages never reside on any server device. While a peer-to-peer processing application has significant advantages over the client-server approach in these applications, agents in these systems must be highly autonomous meanwhile trustful.

Table 3 categorizes the major applications of MAS, with respect to the features of the application domain, specific problems deal with, and features of each type of the applications related to agent characteristics.

It would be desirable to have a statistics on the variations of MAS applications and the major system topology employed in each of the applications. There are two main factors that make it difficult to enumerate the application systems with respect to the topologic types of the agent interactions. One is the limited resource available for the real world MAS applications, especially lacking the application systems with significant influence to the field. The second is that in many real applications, there is no clear cut on which topology the agents in the system apply. More often the applications have a mixture of the interaction topologies among the interactions of the agents in the applications. Instead, we thus turned to a look at the MAS development/construction tools (toolkits, languages, libraries) to find the correspondences of the topology enabled/allowed by these systems/tools. We have evaluated 26 commercial and 39 academic MAS products and/or development packages/toolkits. Tables 4 and 5 summarize the systems. It is found that no any of the above topology is in a dominating position in either domain. However, two observations are worth to mention. One is that while the Star-like topology was seen in 28% of academic systems, there is no (0%) any commercial system adopting this scheme. The other is that the grid-like topology is the most popular one in both the commercial (23%) and academic (36%) systems. Note that quite an amount of systems also possesses the property as a mixture of both grid-like and star-like topology. If we consider this mixture topology together with the grid-like ones, then a majority in both academic and commercial systems is present.

It is not our intention to collect and summarize all published MAS application systems that have been built or reported. Therefore our discussion will be focused on the categories of applications, without referring to specific products or product systems. We thus present an extensive, but not exhaustive, list of work in the field. Despite the youth of the field, space does

Table 3
MAS systems with respect to application domains

Domain of application	Features of the application	Type of agents in need	Suitable topology	Complexity of interaction
Information service	Mixture of distributive and centralized	Diverse	Grid or HCAN	Low
Web search	Distributive	uniform	Web-like	Low
Planning and Scheduling	Centralized, semi-distributive	Heterogeneous	Star-or Net-like	Mild
Process control (manufacture assembly, air traffic)	Semi-distributive, mixture of distributive and centralized	Diverse	Grid or HCAN	High
Reasoning and decision making	Mixture of distributive and centralized	Mixtures	HCAN	high
Data fusion and mining	Centralized	Mixtures	Star or grid or HCAN	mild
Simulation	Mixture of distributive and centralized	Diverse kinds	Star or grid	High
E-commerce	Peer-to-peer	uniform	Web-like	low

Table 4
Commercial MAS development/construction products: total 26

Topology type	Number of systems	Percentage
W	5	19
S	0	0
G	6	23
G/S	6	23
H	0	0
Other	9	35

Star topology: there seemed to be no instances of a star topology in the commercial realm. Because of the size of deployment (load/volume) in a commercial realm vs. academia, that would explain why a star would be deployable in academia, but not in a commercial arena. G/S: the combination of G/S meant that there were options within the framework to allow for either a single entity to perform the controlling function of agents or to distribute that control in a more grid-like pattern. *H* topology: actually found an instance of the Hierarchical in the academic arena. It was described in the product info almost exactly what your paper describes. Other: many commercial products that would probably be classified in the academia world as grid-like, are actually classified as other in commercial because that called themselves a tool to build tools for marketing purposes. In that sense it could be called a particular 'type of topology' but the product information was somewhat confusing.

not permit exhaustive coverage. Instead, the work mentioned is intended to illustrate the techniques that exist to deal with the issues that arise in the various multi-agent scenarios.

5. Example

In the following we present an example design of application of MAS with the four topologies studied in this paper. We know that software agents provide a powerful new method for implementing the next-generation information systems. In the example multi-agent system described below, agents are designed to perform information gathering, categorization, and distribution according to specific needs of users. Special human-system interfaces built in these agents will provide continual support of interactions between IMS and the agents. The hypothetic information service management system must accommodate the following agent assemblies.

Table 5
Academic MAS development/construction products: total 39

Topology type	Number of systems	Percentage
W	2	5
S	11	28
G	14	36
G/S	8	21
H	1	3
Other	3	8

Star topology: there seemed to be no instances of a star topology in the commercial realm. Because of the size of deployment (load/volume) in a commercial realm vs. academia, that would explain why a star would be deployable in academia, but not in a commercial arena. G/S: the combination of G/S meant that there were options within the framework to allow for either a single entity to perform the controlling function of agents or to distribute that control in a more grid-like pattern. *H* topology: actually found an instance of the Hierarchical in the academic arena. It was described in the product info almost exactly what your paper describes. Other: many commercial products that would probably be classified in the academia world as grid-like, are actually classified as other in commercial because that called themselves a tool to build tools for marketing purposes. In that sense it could be called a particular 'type of topology' but the product information was somewhat confusing.

The information service broker agent. The information service broker assembly contains three agents: Publish Service Agent (PSA), Subscribe Service Agent (SSA), and Query Service Agent (QSA). These agents interface directly to the information clients to manage the Pub/Sub/Query Services. The agent functions can be defined as the following.

(1) The PSA possesses the functions of (a) Processing the requests of permission for publish from the publisher (a client), through interactions to I&A (Identification and Authentication) agent. (b) Creating a publisher sequence with the client once permission is granted. (c) Receiving and transmitting the metadata and payload provided by the publisher under a publication request, thereby creating an IO (Information Object) in the IOR (IO Repository). (d) Providing a universally unique identifier (UUID), created by the IOR agent, back to the publisher for future reference.
(2) The Subscribe Service Agent (SSA) will possesses the functions of: (a) Processing the subscriber's requests for permission to subscribe, through interaction to I&A. (b) Processing the subscription predicate (*subscriber metadata constraint*) that the platform applies over the MDR (Metadata Repository) of newly published IOs to determine delivery. (c) Notifying the subscriber of available IOs, generally done thru a client-defined call-back.
(3) The Query Service Agent (QSA) possesses the functions of (a) Processing Query client's requests of permission to query, through interaction to I&A. (b) Informing the Query client to submit a query request containing a query metadata constraint to the platform, once permission is granted. (c) Returning a set of partial result IOs based on the access control policy established for the particular client.

The information management expedition agents. The information management expedition assembly contains the agents for IOR, MDR and I&A management. These agents function as the following.

(1) The IOR agent manages and performs the archiving and organization of published IOs for later retrieval by subscribe and query. The IOR agent is capable of handling a throughput of millions of IOs and hundreds of IO types at a time.
(2) The MDR agent manages and supplies clients with information about available IO types to which the client has access. The MDR contains all schemas and other data for approved IO types and versions within the platform.
(3) The I&A agent associates and ensures a unique identifier with each client/administrator, issues and verifies the authenticator and credentials based on open standards to

the maximum extent with little or no modification of client code.

The information system control agents. The information system control assembly contains the account manage agent (AMA), access control agent (ACA), and persistence adaptation agent (PAA). These agents function as the following.

(1) The Account manage agent (AMA) is responsible for creation of accounts that include issuance of authenticators and credentials; modification of accounts to include disabling accounts, and changing privilege levels via re-issuance of credentials; deletion of accounts.
(2) The Access control agent (ACA) is responsible for granting access to IOs and system resources to authorized clients and administrators. An access control mechanism is enforced by the agent that only allows for the dissemination and receipt of IOs in compliance with the platform access control policy.
(3) The Persistence adaptation agent (PAA) has the capability to manage the lifecycle of information within the platform, ensures interoperability and the system's survival of several generations of clients without degraded service over time. While the IMS (Information manage Staff) is solely responsible for removing information objects from the information space, the PAA provides the means to accomplish this in accordance to policy established.

Thus, the entire exemplar information service management system consists of nine agent modules. In the following, we illustrate the simulative implementation of the information service management agent system in the four topologies, respectively.

5.1. Web-like topological implementation

Note that in this example, agents are classified with different functionalities. However, the interactions among the agents are nevertheless organized in a Web-like topology. This means that every agent in the system is capable of communicating and interacting with each other. The interaction diagram is shown in Fig. 5.

The major advantage of the Web-like topological implementation of the system is that versatile agent functions can be built and incorporated into the system and interaction broadly overall the system. The major problems with this implementation are that (1) it is somehow hard to solve the data inconsistency problem once it happens among the agents, for example, for subscribe service, publishing service, and the IOR maintenance; (2) it is incapable of generating and disseminating user-tailored information under dynamical changes of the situation because adaptation to such a change requires complex coordination of goal and functional specification changes among a number of agents, and (3) the control structure of each agent is rather complicated because of the heterogeneity of the agent modules in the system. Since there is no central controller or mediator, all the control functions among the diverse of agents must be built into each individual agent. We do not recommend such implementation for the supposed information service management system.

5.2. Star-like topological implementation

A Star-like topological implementation of the hypothetic information service management system has the agent interaction diagram as shown in Fig. 6.

In this topological implementation, one extra agent in addition to the nine required agent modules is employed in the system architecture. The additional agent, named Agent

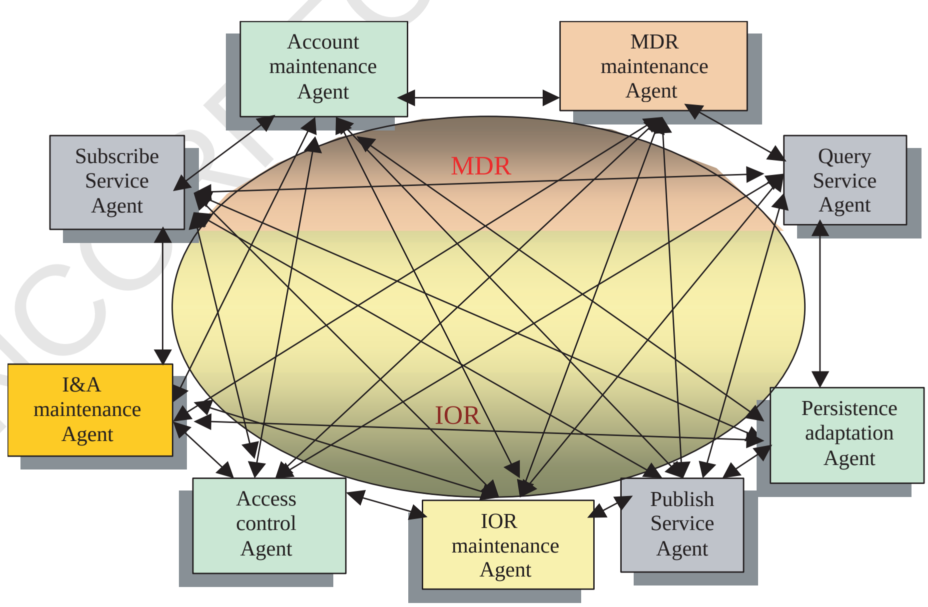

Fig. 5. Web-like topology of MAS for information service management.

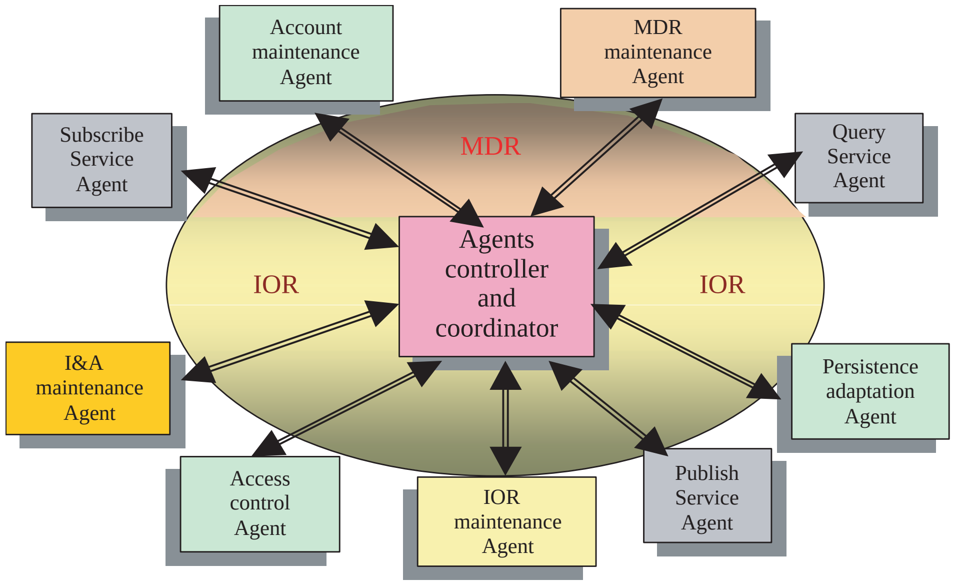

Fig. 6. Star-like topology of MAS for information service management.

Controller and Coordinator, is located in the centralized position among the agents. It has two-way direction connection to all the agent modules, while the information service agents do not directly interact with each other. The advantages of this scheme are that (1) it is easy to solve the data inconsistency problem, and guarantee the right information retrieval and delivery, and (2) it is possible to have additional agents with versatile functions, such as data fusion and mining, added to the service, assuming the agent controller and coordinator maintains properly an agent registry that allows for dynamical addition or deletion of agents in the assembly. Disadvantages of the implementation are (1) it would be less efficient to execute the information retrieval and delivery functions because each of these function requires activation of at least two agents, the coordinate agent and the subscribe or publish agents, and (2) while the control structure of the information service agents will be less complex because each of them only need to interact with the controller, the control structure of the coordinator agent will be relatively complicated. This topological implementation would be a choice if the security and reliability is the main concern and the efficiency (rapid performance of the information service functions) is not a major issue.

5.3. Grid-like topological implementation

In a Grid-like topological implementation, we place the Persistence Adaptation Agent (PAA) at the center of the assembly and the other agent modules surrounding it. However, it differs from the Star-like topology in the way that the other agents all have interactions with their neighboring agents, in addition to the interactions with the PAA. The PAA is chosen sit in the center because its functionality may be need to all the other agents, for example, adjusting the agent functional parameters according to the dynamics of the environment and requirement changes of the system. Here the role of PAA is also different from the Controller and Coordinator agent in the Star-like topology in the way that the PAA does not take the charge of coordinate the execution of the interacting agents. The agents in the system all have certain level of autonomy in terms of performing their designated tasks. The agent interaction diagram is shown in Fig. 7.

Major advantage of this Grid-like topological implementation is that the functionality of the individual agent can be optimally conducted because the agents are connected in the way that only those necessary interactions are permitted. However, this implementation makes it hard to adjust and modify the agent configuration, thus limits the versatility of functions can be incorporated in to the system. The control structure of overall system is also relatively complicated. This implementation thus is also not in our recommendation.

5.4. HCAN implementation for information service management

The design of HCAN architecture and algorithms expedite the integration of publishing, subscribing, and query services in a heterogeneous information space. The system is organized in three agent layers, as shown in Fig. 8: (1) a information service broker layer at the lower level of the hierarchy; (2) a information expedition layer at the middle level of the hierarchy; and (3) a system control layer at the top level of the hierarchy [21]. The functionalities of these layers are described in the following.

The information service broker layer contains subscribe, publish, and query agents to interact with the information service clients and networked information sources, respectively. These agents detect and collect data, perform key word, string, or context extractions from the data feeds, and submit

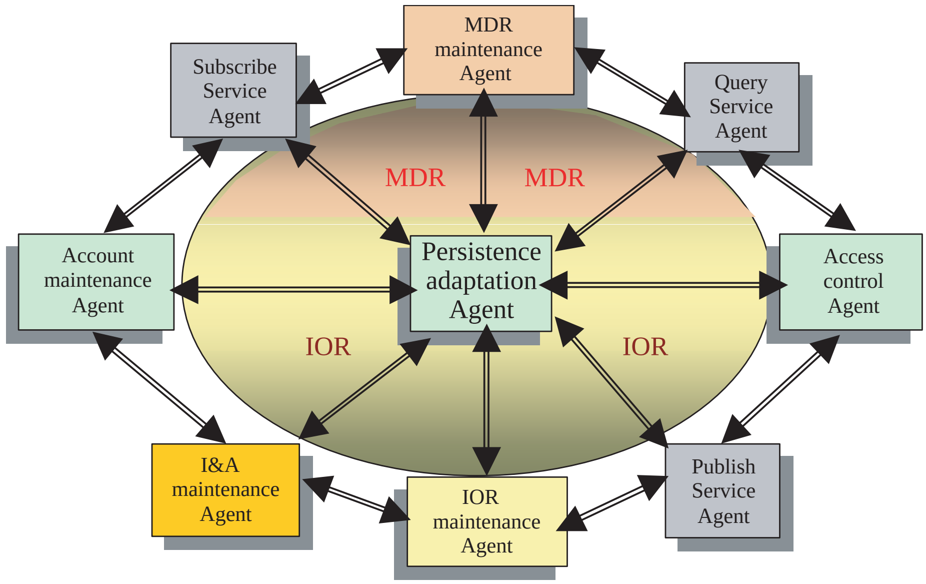

Fig. 7. Grid-like topology of MAS for information service management.

filtered reports to the upper level agents for information package and delivery.

The information expedition layer accommodates three information contents level management agents to perform coordination tasks for information object repository maintenance, metadata repository maintenance, and information source identification and authentication.

The system control layer contains agents to support the information service level management tasks, such as the client account maintenance and access control, and persistence adaptation that performs tasks to adapt the system to environmental variation or requirement changes. The user interface and system management functions are also performed by the management agents at this layer that in charge of interacting with human operators of this information service system.

The advantages of HCAN topological implementation are (1) the agents are better under control of appropriate agents that enables efficiency of each agent's performance meanwhile ensures the reliability of the operations, and (2) the MAS structure is flexible to add additional agents with versatile functions, such as data fusion and mining. Since only agents between layers are connected via heterogeneous links and are interactive, each agent is relatively independent. This makes the additions of agents and modifications of the agent functionalities simple. Major disadvantage of the implementation is that it requires a little more deliberated planning, design, and understanding of the interaction logics of

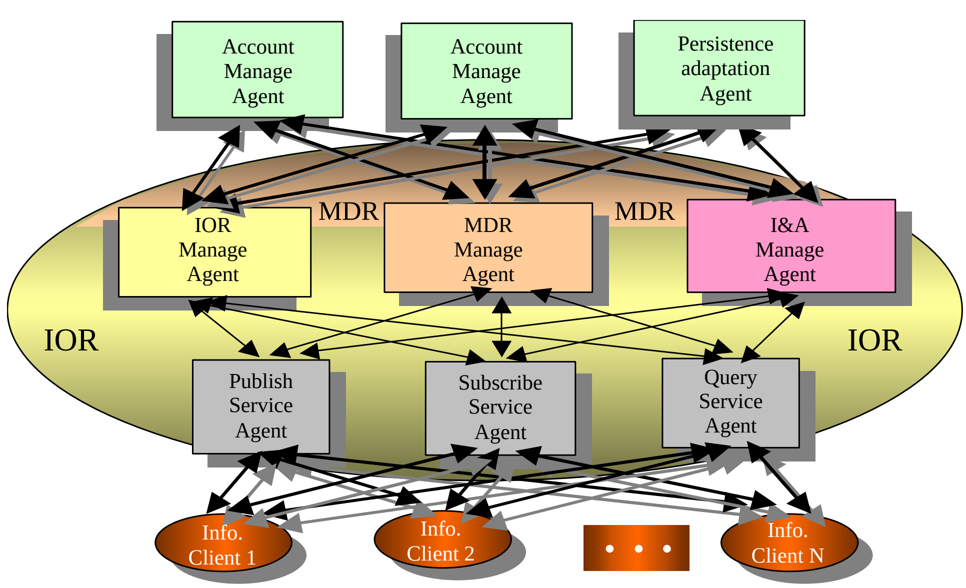

Fig. 8. HCAN topology for information service management.

the agents distributed on different layers. Overall, the HCAN topological implementation is our recommendation for the intended information service management system.

6. Conclusions

The agent-based system developments have emerged from their primarily functional diversities to the stages that raise the necessity of managing the system complexity. Building reliable, maintainable, extensible, and re-usable MASs that conform to their specifications requires modeling techniques that support abstraction, structuring, and modularity. The most widespread methodologies developed for the conventional software systems are various object-oriented approaches. They have achieved a considerable degree of maturity and are supported by a large community of software developers. The system architecture of object-oriented systems is based on the notion of objects, which encapsulate state information as data values and have associated behaviors defined by interfaces describing how to use the state information. Object oriented formal approach address almost all the steps in the process of designing and implementing a software system, providing a uniform paradigm across different system scales and implementation languages. However, there are additional issues related to the development and implementation of multi-agent systems that need to take serious care of.

The implementation of multi-agent systems involves a great number of problems with respect to the components, protocols, interactions, and schemes. In particular it is often hard to guarantee that the specification of a system that has been designed actually fulfils the design requirements. Especially for critical applications, for example in real-time domains, there is a need to prove that the system being designed will have certain properties under certain conditions (assumptions). Many popular multi-agent systems of today deploy agents in a uniform space of operating. The agents are supposed to respond to the same calls and cooperate at the same time toward the goals of operation. That kind of architecture is useful for some applications. However, it endues some difficulties in agent communications and task control. When applied in complex real-time situations with intensive human and system interactions, the cooperative nature makes the system less robust because the disability of one agent would affect the successive operations of the entire agent assembly. In this paper, we studied four major architectural topologies of MAS. The advantages and disadvantages of the topologies are assessed and compared by using a set of criteria based on the functionalities and properties of agents in MAS. The study and understand the MAS topology would help the effort of standardizing agent technology, and hopefully, promote more adoption of MAS in solving real world complex problems.

7. Uncited references

[10], [12], [17], [29].

References

[1] Arai T, Sycara K, Payne T. Experience-based reinforcement learning to acquire effective behavior in a multi-agent domain Proceedings of the 6th pacific rim international conference on artificial intelligence 2000 pp. 125–35.

[2] Barbuceanu M, Fox MS. Cool: a language for describing coordination in multi agent systems Proceedings of the first international conference on multi-agent systems (ICMAS-95).: AAAI press; 1995 pp.17–24.

[3] Bradshaw JM. An introduction to software agents. In: Bradshaw JM, editor. Software agents. Menlo Park, California: AAAI Press; 1997. p. 3–46.

[4] Bradshaw JM, Dutfield S, Benoit P, Woolley JD. Toward an industrial-strength open agent architecture. In: Bradshaw JM, editor. Software agents. Menlo Park, California: AAAI Press; 1997. p. 375–418.

[5] Chauhan D, Baker A. JAFMAS: a multiagent application development system Proceedings of autonomous agents 98. New York: ACM Press; 1998 pp. 100ndash;7.

[6] Cheyer A, Martin D. The open agent architecture. J Auton Agents Multi-Agent Syst 2001;4(1):143–8.

[7] Ciancarini P, Tolksdorf R, Vitali F, Rossi D, Knoche A. Coordinating multiagent applications on the WWW: a reference architecture. IEEE Trans Softw Eng 1998.

[8] Cohen PR, Cheyer AJ, Wang M, Baeg SC. An open agent architecture. In: Oren Etzioni, editor. Proceedings of the AAAI spring symposium series on software agents; 1994, 1994. p. 1–8.

[9] http://www.cougaar.org.

[10] Decker K. Distributed problem solving: a survey. IEEE Trans Syst Man Cybern 1987;17(5):729–40.

[11] Decker K, Sycara K, Williamson M. Middle-agents for the internet Proceedings of the international joint conferences on artificial intelligence (IJCAI-97) 1997.

[12] Ferguson IA, Karakoulas GJ. Multiagent learning and adaptation in an information filtering market Adaptation, coevolution and learning in multiagent systems: papers from the 1996 AAAI spring symposium 1996 pp.28–32.

[13] Flores-Mendez R. Towards a standardization of multi-agent system frameworks, http://www.acm.org/crossroads/crew/roberto_flores-mendez.html.

[14] Geri S, Zhu Q. dbAgent: an intelligent web agent for database mining International conference on computer and informatics (CS I'98) 1998 pp.460–70.

[15] Goldman C, Rosenschein J. Emergent coordination through the use of cooperative state-changing rules Proceedings of the twelfth national conference on artificial intelligence 1994 pp. 408–13.

[16] Giampapa J, Paoluc M, Sycara K. Agent interoperation across multi agent system boundaries Proceedings of agents 2000, Barcelona, Spain, June 3–7 2000.

[17] Grefenstette J, Daley R. Methods for competitive and cooperative co-evolution Adaptation, coevolution and learning in multiagent systems: papers from the 1996 AAAI spring symposium 1996 pp. 45–50.

[18] Haddadi A. Towards a pragmatic theory of interactions Proceedings of the first international conference on multi-agent systems (ICMAS-95) 1995 pp. 133–9.

[19] Hayes-Roth B, Brownston L, van Gent R. Multiagent collaboration in directed improvisation Proceedings of the first international conference on multi-agent systems (ICMAS-95) 1995 pp. 148–54.

[20] Heinze C, Goss S, Josefsson T, Bennett K, Waugh S, Lloyd I, et al. Interchanging agents and humans in military simulation. AI Mag 2002; 23(2):37–47.

[21] Hicks J, Stoyen A, Zhu Q. Intelligent agent-based software architecture for combat performance under overwhelming information inflow and uncertainty Seventh IEEE international conference on engineering of complex computer systems 2001 pp. 200–10.

[22] Lu H, Sterling L. SportsAgents: a mediator-based multi-agent system for cooperative information gathering from the world wide web Proceedings

of the fifth international conference on practical applications of intelligent agents and agent methodology 2000 pp. 331–4.
[23] Lu H, Sterling L. Intelligent matchmaking for information agents cooperation on the world wide web Proceedings of the agent-based simulation workshop 2000 pp. 161–8.
[24] Martin DL, Cheyer AJ, Moran DB. The open agent architecture: a framework for building distributed software systems. Appl Artif Intell 1999;13(1–2):21–128.
[25] Nwana HS, Ndumu DT, Lee LC, Collis JC. ZEUS: a toolkit for building distributed multi-agent systems Proceedings of the third international conference on autonomous agents (Agents'99) 1999 pp. 360–1.
[26] Perry B, Taylor M, Unruh A. Information aggregation and agent interaction patterns in infosleuth Proceedings of CIA 99. New York: ACM press; 1999.
[27] Petrov PV, Zhu Q, Hicks JD, Stoyen, , AD. A hierarchical collective agents network for real-time sensor fusion and decision support The AAAI/KDD/UAI-2002 joint workshop on real-time decision support and diagnosis systems 2002 pp. 73–4.
[28] Rickel J, Johnson WL. Task-oriented collaboration with embodied agents in virtual world. In: Cassell J, Sullivan J, Prevost S, editors. Embodied conversational agents. Cambridge, MA: MIT Press; 2000. p. 95–122.
[29] Sahota MK. Reactive deliberation: An architecture for real-time intelligent control in dynamic environments Proceedings of the twelfth national conference on artificial intelligence 1994 pp. 1303–8.
[30] Shehory O, Kraus S. Task allocation via coalition formation among autonomous agents Proceedings of the fourteenth international joint conference on artificial intelligence 1995 pp. 655–61.
[31] Suzuki N, Fukuda M, Bic LF. Self-migrating threads for multi-agent applications International workshop on cluster computing (IWCC'99) 1999.
[32] Sycara K, Klusch M, Widoff S, Jianguo L. Dynamic service matchmaking among agents in open information environment. ACM GISMOD Rec 1999;28(1):47–53.
[33] Virdhagriswaran S, Osisek D, O'Connor P. Standardizing agent technology. ACM Stand View 1995;3(3):96–101.
[34] White JE. Mobile agents. In: Bradshaw JM, editor. Software agents. Menlo Park, California: AAAI Press; 1997. p. 437–72.
[35] Wilkins DE, Myers KL. A multiagent planning architecture Proceedings of the 1998 international conference on AI planning systems, Pittsburgh 1998 pp. 154–62.