Vira Smirnova

MULTI-AGENT SYSTEM FOR DISTRIBUTED DATA FUSION IN PEER-TO-PEER ENVIRONMENT

Master's Thesis

Mobile Computing

28.11.2002

University of Jyväskylä

Department of Mathematical Information Technology

Author: Vira Smirnova

Contact information: Roninmäentie 1G 2c, 40500 Jyväskylä, vsmirno@cc.jyu.fi +358(41) 502 5767

Title: Multi-agent System for Distributed Data Fusion in Peer-to-Peer Environment

Work: Master's Thesis

Number of pages: 89

Study Program: Mobile Computing

Keywords: Peer-to-Peer, Multi-agent systems, Data fusion, Distributed Sensor Networks

Abstract: This thesis work was aimed to design a new model for fault-tolerant and high availability of Distributed Sensor Networks and data fusion. The main problem was to modify existing data fusion process into a new one by combining multi-agent systems with P2P environment. In the research process was developed Peer-to-Peer distributed Sensor Network model and described a system using components and agents with several scenarios and states. The proposed design also considers the behaviour of the system in common failure conditions. Finally a comparison between DSN, MADSN and P2PDSN was formed using general metrics used for analysing complex distributed systems.

Terms and Abbreviations

AA	Analyser Agent
AI	Artificial Intelligence
BS	Base Station
CA	Controller Agent
CAS	Complex Adaptive System
CDF	Centralized Data Fusion
CORBA	Common Object Request Broker Architecture
DARPA	Defense Advanced Research Projects Agency
DDF	Decentralized Data Fusion
DNS	Domain Name Service
DSN	Distributed Sensor Network Model
DSNs	Distributed Sensor Networks
IP	Internet Protocol
MAS	Multi Agent System
NTP	Network Time Protocol
P2P	Peer-to-Peer
P2PDSN	Peer-to-Peer Distributed Sensor Network
PE	Processing Element
RPC	Remote Procedure Call
SA	Sensor Agent
TCP	Transport Control Protocol
UDP	User Datagram Protocol
VA	Viewer Agent

Contents

1 Introduction

1.1 Background

As more AI applications are being formulated in terms of spatially, functionally, or temporally distributed processing, multi-agent systems (or what was previously called distributed AI) are emerging as an important subdiscipline of AI. This is especially true as the outlines of a potential model for computing in this century are beginning to coalesce: a model in which networks of interacting, real-time, intelligent agents could seamlessly integrate man and machine. Agents in these networks need to be highly adaptive due to their "open" operating environments, where the configuration and capabilities of other agents and network resources can change dynamically. Agents in such environments aim to produce the best possible result given their available processing, communication, and information resources.

In general, multi-agent systems (MAS) are computational systems in which several autonomous agents interact or work to perform some set of tasks or some set of goals. These systems involve computational agents that may be homogeneous or heterogeneous and may involve with common or distinctive goals. Research and practice on systems generally focus on problem of communication, and coordination aspects, as distinct from low-level parallelization or synchronization issues focusing more on distributed design, implementation, and assessment of multi-agent systems raise specific issues. These include coordination strategies that enable groups of agents to solve problems effectively; negotiation mechanisms that serve to bring collection of agents to an acceptable techniques for conflict detection and resolution; protocols by which agents may communicate and reason about interagent communications; and mechanisms whereby agents can maintain autonomy while still contributing to overall system effectiveness.

The need to interact in such systems arises because agents are solving sub-problem that are interdependent, either through contention for resources or through relationships among the sub-problems' goals. For agents to achieve compatible (non-conflicting) and "optimal" solutions to their interdependent sub-problems with minimum use of resources requires them to have sufficiently current, complete, and consistent views of the overall problem(s) and of one another.

Obtaining this information is often not practical due to:

a) Limited communication bandwidth and the computational costs of packaging and assimilating communicated information.
b) The heterogeneity of agents, which makes it difficult to share information, and the potential for competitive agents who, for their own interest, are not willing to share certain information.
c) The dynamic character of the environment due to changing problems, agents, and resources, and the inability to predict with certainty the outcome of agents' actions.

MAS systems are a new promising paradigm in computing, which contributes to various fields including Distributed Sensor Networks (DSNs). DSNs are networks with sensors in different locations and different types of nodes for recording, analyzing and monitoring various parameters from the environment. DSNs can be used in different areas such as:

- Traditional platforms: ships, subs, planes, tanks, vehicles.
- Environmental monitoring: air, sea, and ground.
- Machinery monitoring, vehicle tracking, smart surfaces.
- Surveillance, security, detection, command views.
- Chemical, biological, and nuclear weapon detection.
- Human and animal bio monitoring and etc.

By combining sensors from different positions or with different frequency ranges measurement accuracy is improved. The process of collecting and analyzing data is called *data fusion* that improves targeting.

In a few last years a very promising concept has risen called Peer-to-Peer (P2P) technologies. Being massively successful for end-users file-exchange architectures, researchers have focused in the P2P development to provide new specific architectures of which a cloud of peers can share any number and any type of resources and for collaborative work. Consequently we have undertaken the design of an agent-based P2P system for distributed sensor network.

P2P and agent-based systems are well suited for this task as explained in the thesis further. Our first main design decision was to make a system following the P2P paradigms of networking as opposed to the traditional client-server model. It avoids the single point of failure, which is a major weakness in large-scale distributed systems. This also means that removing any node from the network does not degrade the performance of the system critically.

Agent-based systems offer the desired characteristics to implement a P2PDSN system. Indeed, from a design point of view agent abstraction represents the behavior of peers because of their autonomous and social natures. In such P2P context their complex interactions can be the basis for a cooperative multi-agent system. Additionally, agent systems support ontologies able to provide extensible descriptions of resources, which are fundamental characteristic required for building an open distributed system.

1.2 Related Work

There are quite many projects that are involved in researching Distributed Sensor Networks and Multi-agent systems separately. But applying agents for P2P networks is a new sector that has just in the near past started to be researched. That means that not so much work has been done in that direction. But any way there are some of them that I found.

The most vivid example is James E. Youll who made a research work from Massachusetts Institute of Technology. He developed a model “Peer to Peer Transactions in Agent-mediated Electronic Commerce” using the same approach this thesis has, but in a different sector his being electronic commerce. In his work he made also some practical tests showing a possibility of applying intelligent agents to P2P environment [Y01].

A research work applying agents for DSNs has been done in USA at the Advanced Technology Laboratories [ATL] is an advanced-computing asset of Lockheed Martin Corporation – a global enterprise with core businesses in systems integration, space, aeronautics, and technology services. Lockheed Martin is the nation's largest defence contractor. By developing and applying computing innovations in artificial intelligence, distributed processing and embedded processing. The most related work to mine is the data fusion for supporting situations awareness on the digital battlefield where was considered a mobile agent model for data fusion in P2PDSN [J00].

Another research project supported by DARPA [DARPA] is located in University of California at Los Angeles. Their topic is “A Framework for Efficient and Programmable Sensor Networks” [BS01]. Their research based on a real implementation suggests applying mobile agents for wireless sensor networks. A possibility of applying mobile agents to DSN was also considered in one more project Multi-Resolution Data Fusion using Agent-Bearing Sensors in Hierarchically-Organized Sensor Networks (MU-FASHION [MUFASION] where homogeneous sensors from which mobile agents collect data and make a partial analysis of it was tested. The suggested approach is an extension of general DSN called MADSN (Mobile-agent based DSN) which I will introduce more thoroughly in the thesis with a comparison in chapters 5 and 9.

1.3 Research Problem Statement

Our research problem was to enhance existing data fusion models to incorporate also the characteristics of fault-tolerance and high-availability thus providing a basis for robust system design.

The research project I worked for is called Cheese Factory [CFP] that studies peer-to-peer communication and behavior of peer-to-peer networks concentrating mainly on distributed search of resources and their efficient use. The project is developing Chedar – distributed Peer-to-Peer computing and storage system that can be used as a platform for distributed applications. As research goals for Cheese Factory the following problems are aimed to be investigated:

- formation and search in power law networks,
- optimisation of network load, organisation and management in peer-to-peer networks,
- efficient matching of logical and physical topology,
- peer-to-peer network resilience under attacks and random failures

- application prototypes and software architectures suitable for peer-to-peer middleware services and
- real-time and decentralized search in grid computing systems such as Globus Toolkit [GT]

1.4 Structure of Thesis

The thesis presents an agent-oriented Peer-to-Peer DSN model including a MAS with data fusion and P2P technologies. The developed model was is compared between two other models found from research literature.

The thesis is divided into 10 chapters. In the beginning Multi-agent systems are introduced (Chapter 2) in which two basic paradigms such as AOIE (Agent, Organization, Interaction and Environment) for description of a whole system and PAGE paradigm for making a full review of agents in the system (Chapter 3) are pointed out. A MAS methodology for designing a multi-agent systems is presented in Chapter 4.

Next the problem domain of Distributed Sensor Networks and data fusion are presented (Chapter 5). This chapter provides the knowledge about DSN and MADSN models. Chapter 6 starts the practical part of the thesis by introducing Peer-to-Peer Distributed Sensor Network (P2PDSN) model. A system for P2PDSN is developed step-by-step using AOIE and PAGE paradigms in Chapter 7 and different states of the system illustrated in Chapter 8.

Finally a comparison between Distributed Sensor Network (DSN), Mobile Agent-based Distributed Sensor Network (MADSN) and Peer-to-Peer Distributed Sensor Network (P2PDSN) models is shown in Chapter 9 and the thesis concluded in Chapter 10.

2 Multi-Agent Systems

Multi-agent systems (MAS) are collections of several computational entities, called agents, interacting with one another. Autonomous agents can observe their environment and perform simple local computations leading to actions based on these observations. The behavior of an agent may be non-deterministic and its actions may modify the environment as well as the agent's location within the environment. What distinguishes MAS from other agent models is that there is no central coordination of activity. In the context of MAS, emergent behavior manifests itself as swarm intelligence whereby the collection of simple agents of limited individual capabilities achieves "intelligent" collective behavior [WP99].

The use of agents allows distributing tasks among them each reacting to the environment's changes and applying its individual knowledge and skills for the decisions. Each of agents has its own percept and following the goals he has uses different kind of algorithms. Therefore an agent is relatively autonomous; it has a considerable degree of freedom in how it interacts with other computational and human agents. Agents can communicate, cooperate, coordinate, and negotiate with one another, to advance both their individual goals and the good (or otherwise) of the overall system in which they are situated. Agent societies can be structured and mechanisms instituted to encourage particular kinds of interactions among the agents. Populations of agents acting on their individual perspectives can converge to systemic properties. Teams of agents, each providing a particular suite of capabilities needed by others, can be constructed and deployed to collectively solve problems that are beyond their individual abilities. This teaming can even be done on the fly, and can include humans as well as heterogeneous computational agents [D98]. In general MAS is a multi-disciplinary research area including such sciences as physical, computational, natural, economical, social and life sciences.

2.1 Main Characteristics of MAS

Agents are the main entities of the MAS. The MAS paradigm is based on the existence of these agents having goals to reach and communication capabilities to interact one with another. An agent has mechanisms to act according to certain situations to reach its goals. The whole picture about agents will be presented in section 0.

MAS can be characterized such that:

1. each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint,
2. there is no global control in the system,
3. location of data is decentralized and
4. computation is asynchronous.

There are many areas that MASes can be applied and next some of the problem domains will be introduced. Multi-agent systems can be used to solve problems that are too large for a centralized agent to solve because of resource limitations or the sheer risk of having one centralized system that could be a performance bottleneck or could fail at critical times. The interconnection and interoperation of multiple existing legacy systems can also be done using MAS. To keep pace with changing business needs, legacy systems must periodically be updated. Completely rewriting such software tends to be prohibitively expensive and is often simply impossible. Therefore, in the short to medium term, the only way that such legacy systems can remain useful is to incorporate them into a wider cooperating agent community in which they can be exploited by other pieces of software. Incorporating legacy systems into an agent society can be done, for example, by building an agent wrapper around the software to enable it to interoperate with other systems [GK94].

Third area MASes have been used is to provide solutions to problems that can naturally be regarded as a society of autonomous interacting components-agents. For example, in meeting scheduling, a scheduling agent that manages the calendar of its user can be regarded as autonomous and interacting with other similar agents that manage calendars of different users [GS96]. Such agents can also be customized to reflect the preferences and constraints of their users. Other examples include air-traffic control [CGS98] and multi-agent bargaining for buying and selling goods on the Internet.

MAS can provide solutions that efficiently use information sources that are spatially distributed. Examples of such domains include sensor networks [PIKM99] in which this thesis also mainly concentrates on.

The performance enhancements usually associated to MASes can be found from the dimensions of:

1. *computational efficiency,* because concurrency of computation is exploited (as long as communication is kept minimal, for example, by transmitting highlevel information and results rather than lowlevel data),

2. *reliability*, that is, graceful recovery of component failures, because agents with redundant capabilities or appropriate interagent coordination are found dynamically (for example, taking up responsibilities of agents that fail),

3. *extensibility,* because the number and the capabilities of agents working on a problem can be altered,

4. *robustness*, the system's ability to tolerate uncertainty, because suitable information is exchanged among agents,

5. *maintainability,* because a system composed of multiple components-agents is easier to maintain because of its modularity,

6. *responsiveness,* because modularity can handle anomalies locally and does not propagate them to the whole system,

7. *flexibility,* because agents with different abilities can adaptively organize to solve the current problem, and
8. *reuse,* because functionally specific agents can be reused in different agent teams to solve different problems.

2.2 Areas of Research in Agent Systems

The MAS concept comes from two different influences, which are the systemic, and the Artificial Intelligence (AI).

The *systemic* is like macro view of a problem and concentrates on the flow of information between components. The systemic is interesting, as it has developed many of its ideas in terms of interactions and communications.

AI is the discipline, which introduces the concept of cognition within processes. Different techniques such as neural networks and expert systems have been developed to allow for a dynamic choice of actions according to certain situations. AI techniques have been heavily used for control and diagnostics issues.

The problems of such paradigms come from the difficulty in representing different behaviors and different interactions between the entities, which compose a general system. The systemic considers the macroscopic view of a problem and flow of information but does not help in describing behaviors and the traditional AI approach is very centralized and does not consider interactions.

For the model presented in the thesis I have chosen the systemic view of research because it suited well to our Peer-to Peer systems research approach and project’s goals.

Multi-agent systems were originally developed to artificially simulate systems in different natural, life, economical and social sciences, where it is necessary to distinguish the component behaviors and to find a way to enrich communication capabilities (perception, information or order sending). Such kind of applications are more and more used either for task distribution and resource allocation or for specific applications where components are heterogeneous.

The MAS research community currently determines four main areas of work [SD01]:

- The specification and understanding of the context in which a multi-agent system executes: how the languages, protocols, organizational structures, goals, and incentives influence the kinds of interactions among agents. This covers: communication languages and protocols (semantics, pragmatics), organization and social structure (agent roles, social laws), co-operative problem solving (collective goals driving individual choices), decentralized systems (individual choices driving collective behavior), and mechanism design (incentives for aligning individual and group goals).
- Techniques that agents use to reason about the multi-agent system: conflict resolution and negotiation, multi-agent planning [STX00], coalition formation and organization self-design [MB98], agent modeling and plan recognition, multi-agent learning [BW98], distributed search [JNFOO00, BM01] and constraint satisfaction, and foundations (multi-agent logics, game-theory, economics, philosophy).
- The fundamental reasoning techniques coupled with the multi-agent context need to be implemented and tested to evaluate the strengths and weaknesses of the theoretical underpinnings, as well as to provide practical tools for developing complex software systems. This includes: agent programming languages [B98, FG98], multi-agent programming frameworks, agent models and architectures, standards for multi-agent technology (interaction protocols, languages), development and engineering methodologies, evaluation of multi-agent systems, testbeds and development environments, and user interfaces and personalisable agents.

- Applications of multi-agent systems provide touchstones for measuring progress as well as illuminating important, previously overlooked problems that arise in the real world. The ICMAS'98 call covered the following domains: electronic commerce [GGRG98], cooperative information systems [AD98], distributed resource allocation, information agents on the internet [IUI98], multi-agent simulations of social and biological systems [PC98], multi-agent vision and robotics, believable agents in multi-agent settings, and interacting personal digital assistants.

The area resembling most of the work done in Cheese Factory Project [CFP] is the second one: techniques that agents use to reason about the MAS. In particular our Chedar platform is used for distributed search providing means for agents to locate resources and services.

While the research community is too large and diverse to agree on a particular methodology, it is often the case that well-balanced research activities span several of the above-mentioned topics. That is, investigations into aspects of multi-agent systems often tie together ideas on the nature of the agent interactions, how the computational agents should operate within this framework, how the results have been developed and evaluated, and the practical significance of the work. Because of the diversity I will not go further about the discussion of different research methods but introduce the ones used in my research work in next two chapters.

3 AOIE and PAGE Paradigms

In the next part the four main components of a MAS will be described: Agent, Environment, Interactions and Organization, as they have been identified in [D95]. This section is aimed on describing briefly the AEIO paradigm that was used by the AGENT-project [INPG98]. This approach promotes that a multi-agent system can be decomposed according to four components. The first component is the Agent (A) as the basic component of a multi-agent system; the second one is the interaction (I) which ranges from very simple ones like physical forces exerted between agents. The third component is the Organization (O) which is often identified as the entire multi-agent system or the society of agents, but which may clearly be exhibited as an independent feature of MAS [D97]. The last component is the Environment (E), in which the agents evolve; this environment can be physically or virtually modeled according to choices of software agents.

3.1 A as Agents

There is a great amount of definitions of “agent”. However for the thesis work I have selected only the ones that emphasize different aspects of agents.

Definition 1. *“An **agent** is anything that can be viewed as **perceiving** its environment through **sensors** and **acting** upon that environment through **effectors**”.* [RN95]

This first definition points out that the most important for agent is its environment and how information can be delivered to agent about its changes. Like a human agent should have eyes, ears and other organs for sensors and hands, legs, mouth and other body parts for effectors.

Definition 2. *"Agent is a sophisticated entity acting rationally and intentionally, either isolated or in cooperation with other agents."* [S91]

Definition 2 was found in a dictionary for AI terminology. According to their definition, an agent is a rather complex entity since it has to act rationally and intentionally. What rational behavior means is clearly dependent on the application domain so agents seem to be closely connected to an application area. The choice of the word intentionally is interesting since it represents a branch of agent research where agents are viewed as intentional systems, whose behavior can be predicted and explained in terms of attitudes such as beliefs, desires, and intentions. The approach has been successful since it abstracts away from the implementation by mimicking well-known human behaviors. As in most definitions, agents are allowed to exist both in isolation and in an environment inhabited by other entities. In this definition, agents cooperate with other agents, which requires communication mechanisms (languages and protocols).

Definition 3. *"Agent is a software entity which functions continuously and autonomously in a particular environment, often inhabited by other agents and processes."* [S97]

This definition stresses the independence of agents. They must be able to function autonomously. In addition, the agents should function without interruptions or manual interventions (continuous operation).

Definition 4. *"Intelligent agents continuously perform three functions: perception of dynamic conditions in the environment; action to affect conditions in the environment; and reasoning to interpret perceptions, solve problems, draw inferences, and determine actions."* [H95]

Agents are long-lived systems that operate *continuously*. Therefore agents require support for persistent execution. The combination of perception and action is often called reactivity.

3.1.1 Different Types of Agents

Agents possess many properties, but what are the real properties for each agent depends on its type. Classification of different types of agents can be divided into 5 main categories:

- Classical rule-based agents,
- Reactive agents,
- Hybrid agents,
- Learning Agents and
- Mobile agents.

Classical rule-based agent has a symbolic representation of the world that can be knowledge about its environment (beliefs, desires, intentions), rules that apply in the world and also its goals. “Intelligence” comes from symbolic reasoning. The brightest classical type of such agent is a planning agent of which architecture is shown on the Figure 3-1.

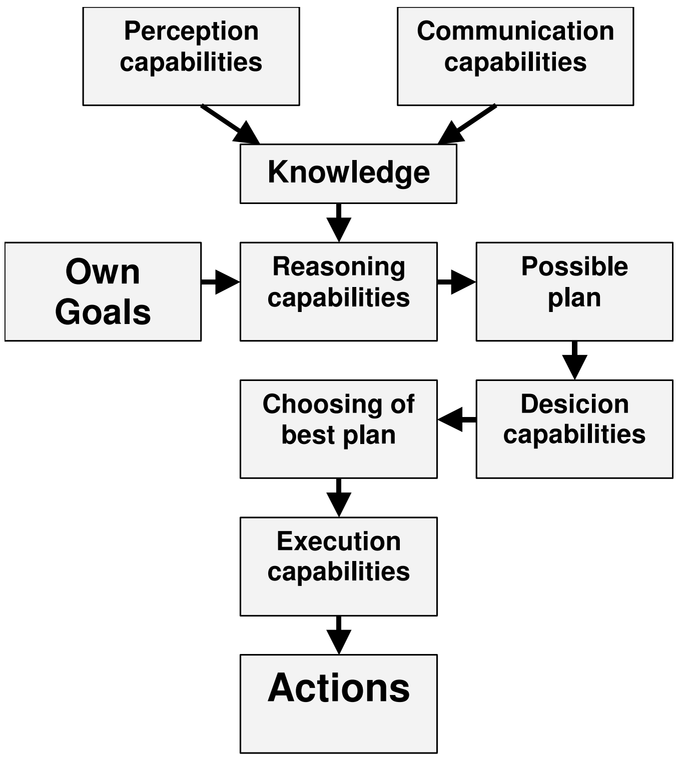

Figure 3-1. Planning Architecture of Intelligent Agent

In *reactive agents* an intelligence is a product of interaction between an agent and its environment. There is no symbolic representation or reasoning. An agent is a collection of task accomplishing behaviors where each behavior continually maps perceptual input to action output. Overall behavior is determined by interactions between behaviors.

Hybrid agent architecture mixes the good ideas of the previous approaches. Hybrid agent consists of several subsystems:

- subsystems that develop plans and make decision using symbolic reasoning
- reactive subsystems that are able to react quickly to events without complex reasoning

Because there are several subsystems such type of agents need to have a controlling system.

Without learning there is no intelligence so therefore one type of agents represents learning aspect. *Learning agents* can use several learning algorithms making it a main problem how to apply them for right purposes. Using of learning components improves the capabilities of previous agent types.

The last types of agents are Mobile agents *that can move between agent servers. It is reasonable to move an intelligent agent:*

- to a more powerful machine to do computationally heavy tasks like learning,
- to where it is needed and
- to move the agent before connection is lost to keep it working.

3.1.2 Properties of Agents

Readings about agents [HS97] contains a massive overview of 44 agent properties. But instead of listing all these properties all of them were classified and catalogued according to aspect and occurrence frequency in order to obtain the following list resembling the most used ones:

1. Autonomous
2. Cooperative (communicate)
3. Reactive
4. Cognitive
5. Adaptive
6. Software
7. Intentional (beliefs, goals, desires)

8. Deliberative (reasoning)

9. Longlived (persistent, continuous)

10. Isolated

11. Mobile

12. Heterogeneous

13. Learning

14. User dialog

15. Reliable

16. Collaborative with user

17. Rational

Going through all these properties would require too much space, so I picked up some of them that are closely related to the model developed hoping to illustrate the idea between different properties.

Autonomous: An agent has goals to reach and mechanisms to act in order to reach those goals. This aspect defines what is called as *autonomy*. If no supervisor chooses for the agent what it should do, however, we will see hereafter that an agent can be an underlying when it receives orders from others entities and when it accepts such orders: some agents can refuse whilst other cannot, due to their possible roles in the organization. Finally, the autonomy of an agent is not only related to its behaviors but also to the availability of the resources it needs to exist and behave within the global system.

Cooperative: An agent can also communicate with other agents in different ways. In order to communicate it needs perception mechanisms to know whom he can communicate with. Agents can ignore, cooperate, or compete with each other. Most common is agent cooperation where agents combine their efforts in achieving a common goal. Cooperation is closely associated with intentional behavior since the agents need to be aware of other agents' goals, beliefs, and desires. Strategies proposed for cooperation include fixed cooperation strategies, teamwork learning, hierarchical organizations, etc. Inter-agent communication is vital for cooperation. Generally, the communication is at an abstract level and based on agent communication languages and protocols. More generally, an agent needs to have some actions, which external effects are perceived as behaviors (methods in the Object-Oriented paradigm). If an agent exhibits different behaviors it needs to have a mechanism to choose the best mechanism to use and the mechanism of choice is the real intelligence of the agent. Two different kinds of choice capacities, which make the difference between *reactive* and *cognitive* agents is usually distinguished. Though, despite the separation, the distinction is not always clear.

Reactive: The term reactive is used in the agent literature in at least two different meanings. One is where reactive refers to an agent's ability to react on the environment in a timely manner. In the other, reactive refers to agents' actions that are taken without consulting an internal model. Wooldridge-Jenning [FG99] use the term reactive for agents that perceive their environment, and responses in a timely fashion to changes that occur in the environment. Russel and Norvig [RN95] refers to reactive when agents can use reactive planning and when agent use condition-action rules to define their behavior.

Cognitive: A *cognitive* agent reasons. It also owns a knowledge base and also needs to obtain information about itself and other agents to act. The difference is that it is able to act even if the case it perceives is not already described in its knowledge base. It can also foresee what it should do next (planning capacities) and it is able to learn from its own experience or from the experience of others. To reason a cognitive agent often needs explicit representations of itself and of a part of the environment and more specially a representation of the agents with which it communicates.

Most of the progresses made in MAS with regards to cognitive agents have been performed in close connection with social and human sciences. In essence the main difference between the two kinds of agent is that the cognitive agent has much more flexibility in terms of its behaviors, as it is able to compare, to foresee and to learn. But it clearly needs more information and involves harder design to be built so that the real key is to always adapt the complexity of the choice of the agent to the only needs of the problem to be solved.

Adaptive: An agent is a system that tries to fulfill a set of goals in a complex, dynamic environment. An agent is called autonomous if it operates completely on its own, deciding by itself what action it should take, considering the input coming from its sensors, in order to achieve its goals. An agent capable of improvements at achieving its goals is said to be adaptive.

Software: A fourth of the definitions explicitly state that agents are software. In some definitions, also hardware devices and humans are allowed as agents although a majority leaves the physical nature of the agent undefined.

Intentional: When agents can be viewed as entities dealing with mental components such as beliefs, goals, intentions, desires and so on, then the literature views them as intentional. Russel and Norvig [RN95] refers to intentional states as internal states of the agent that deals with believing, knowing, desiring, fearing, and so on, about some aspect of the external world.

Deliberative: Terms like deliberative or reasoning are used to describe the agent's ability to reasoning. It is used when agents use knowledge, and some reasoning to define what action to take. [HS97] views an agent's level of cognition as a range between reactive to deliberative.

Persistent: Terms like persistency, continuous operation, long-lived operation, are used for describing the lifespan of an agent. Some agents' definitions state that it is mandatory for an agent to be long-lived.

3.2 O as Organization

An organization means that there is a sort of unity between agents that form a group. Whenever a group can be represented explicitly, it will be called an organization. Implicit groups may exist from the point of view of a single agent whilst neither the other agents nor the user may be aware about them [VD99], but as they can be represented by the agent itself explicitly, they will be considered as organizations too. If we limit our definition of organization here, it should be clear that the notion of organization has several meanings in the MAS domain. As an example of an alternative and more general definition, an organization can be defined as a set of agents performing roles and interacting along organizational links [DF98].

In this simple framework of what organizations are, groups can be defined externally (prior to any process) according to external knowledge (i.e. exogene organizations) or can *emerge* from specific state of a group of agents (i.e. endogene organizations). So it means that organization building mechanism can be top-down (from system specifications) or bottom-up (from the agent).

An organization has different functions in a MAS:

- It can be used to represent more global goals and to realize global actions. In such case an organization is an agent, has its own goals and acts and interacts to reach its goals.

- It can be used to help agents to perform its tasks or to allocate the resources for agents. The organization can change agents' goals or give them information, for instance by increasing agents perceptive abilities, or give them orders.

- It can be used to control agents. As local and more global goals are not always compatible, an organization provides an overview to check if agents' actions respect the global goals or not.

3.3 I as Interactions

The idea of communication between agents is not only transferring bits through channel, it means to reach the high-level interoperability via communication! Interactions consist of communication language (usually protocol) and decisions (reasoning what primitive to use). This chapter is shown further by *communication mechanisms* depending from *types of interaction*.

3.3.1 Agent Communication Mechanisms

The development of agent communication mechanisms via Agent Communication Languages is underway, however, the major languages have either no provision or no well-defined semantics for group communication, which is often needed. To give the idea of current state list of the most often used ones is described next.

Knowledge Query and Manipulation Language (KQML) and Knowledge Interchange Format (KIF)

KIF [GF92] is an extended form of first order logic for encoding the content of a knowledge base. KQML is a language to support communication of KIF-like constructs among software agents [FLM97]. The language was designed by a consortium called the ARPA Knowledge Sharing Effort (KSE) [NFF91], a project to create infrastructures for sharing and reuse of knowledge bases and the systems around them. It is implemented not by the standards body but by independent research labs building KQML compatible systems. The argument remains that efforts to bind recorded knowledge to formal structures may be premature. Ginsberg [GM93] points out that a contemporary language could be fitted for compatibility with all existing knowledge representation (KR) methods, but that it could not possibly anticipate every form of KR developed in the future.

He cautions that even mildly constraining KR standards could stifle development of non-conforming approaches. Ginsberg also suggests that research into formal KR methods should be properly labeled *research* not *standards* to preclude confusion, and that the true research focus should be "whether knowledge sharing is possible at all."

Foundation for Intelligent Physical Agents (FIPA)

FIPA [FIPA] is a membership-based organization that builds and endorses specifications "for the interoperation of heterogeneous software agents." The FIPA repository of catalogs the myriad FIPA standards including agent communication, message transport and management, architectures, and applications. A complaint about FIPA may be simply that there's too much of it. Rather than considering the design of systems that adaptively learn to communicate with one another, FIPA seeks to hardwire every aspect of inter-agent activity from message structure to management.

Simple Object Access Protocol (SOAP)

SOAP is an XML-based description of computer-to-computer message passing. It also describes optional remote procedure calls and message transmission via HTTP, the primary (and often only) bearer of SOAP messages. Messages between agents in the Atomic Market or similar systems could be encapsulated in a SOAP envelope, although they are not at present. Low level Atomic Market communications use queued, asynchronous messaging. The agents could be adapted to use SOAP's HTTP transport, though asynchronous e-mail-like implementations with external queuing seem to be better suited to the Atomic Market architecture, as some sort of on- or off-agent queue is inevitable. Current implementations of SOAP define only HTTP messaging, though mappings for other transports, including SMTP, are anticipated [B01].

Web Service Description Language (WSDL)

WSDL describes an XML document to define how to get a web-based service, the operations supported by the service, the parameters the service expects, and the values/structures the service returns [W01]. WSDL is a product of the UDDI/SOAP working group, and is used in conjunction with SOAP to connect independent processes to remote web-based services.

In the MAS community, it is admitted that the term communication in MAS means more than it means in traditional Distributed Systems [CDK01]. Unfortunately, whereas communication already is standardized in the latter, there is not yet a common agreement about how communication should be treated in MAS. Some attempt has been made in the framework of KQML [FLM97], and now the work is continuing internationally in the FIPA organization. Still there are many open questions like how rich the syntax of a message should be to aid its fast interpretation? Is it possible to define application independent protocols? If there exists a set of primitives can one build protocols from these primitives? The answers to these questions have to be found at the MAS level and not only at the Distributed System level that might implement part of the communication between agents. To distinguish the MAS work from the standardized communication issues that exist in Distributed Systems, and for discussing communication issues between agents, usually people refer to *interaction* rather than to communication.

3.3.2 Types of Interactions

Interactions are dynamic relationships between agents. These relationships are the results of a set of agent actions. The type of interaction between agents depends on:

- their goals: Agents' goals can be compatible or not. If goals are compatible, agent can *cooperate* ,if not there is *competition.*

- their capabilities: An agent's behaviors, alone, are not sufficient to allow it to reach its goals. In this case, the agent must interact with others to find help.
- their resource allocation: Are agents in competition with other agents to realize their own tasks? Among resources there is CPU, space, time schedule, access to information, etc. Conflicts occur when a set of agents needs the same resources at the same time. To manage or avoid such conflicts, agents need to *co-ordinate* their actions. Three principles of co-ordination exist: the negotiation in between agents by means of information exchange, the arbitration where an upper level agent chooses for the agents what should be done, the reactive co-ordination through the environment, as usually performed by reactive agents but not only them [DF98]

The interactions between agents can be either direct (agents communicate one to another) or indirect, by means of the environment: the consequence of an agent's acts of any kind changes the environment thus becoming different for the other agents. Some groups of agents, that will be described hereafter, are the results of agents' specific interactions.

3.4 E as Environment

Every thing that is not inside agents characterizes the environment of the agent. The environment represents the space where all agents live. For robots, the Environment represents the Euclidean space in which agents are moving. For software agents, the computer network might represent the environment. Whenever agents are located, the environment is often the metric space when available or easily defined. Obviously, if the environment of the agents refers to the geographical space, it will be the case. But that is not always easy to determine, as in telecommunications domain for example [VD99]. The agent needs a certain representation of its environment, even if it is not the entire environment. At least, it needs to be able to locate itself within the environment. The environment of an agent both defines and limits the capacity of interaction of an agent towards the rest of the system.

3.5 PAGE Paradigm

Before designing any agent from Multi-agent system it is important to define each agents' percepts and actions, what goals or performance measure agent is supposed to achieve and what sort of an environment it operates in. For doing these [RN95] suggests to use so called PAGE-table meaning Percept, Action, Goal, and Environment descriptions.

Percepts can be any kind of sensors (or inputs), which are fixing the current situation in the environment. Each agent has his own perceptions. Also in the role of perception can be the communication primitives that agents get in a process of interoperability. Important part of agent's architecture is the *goals* it possesses. Using a list of *actions* and new updated perceptions agent acts according to these goals.

After each action has been made agent evaluates its so-called "coefficient of happiness" that shows how the goal was reasonably reached. These actions reflect to the environment through the effectors, which also can be called as outputs. How all the main components of PAGE paradigm interact is shown on the Figure 3-1

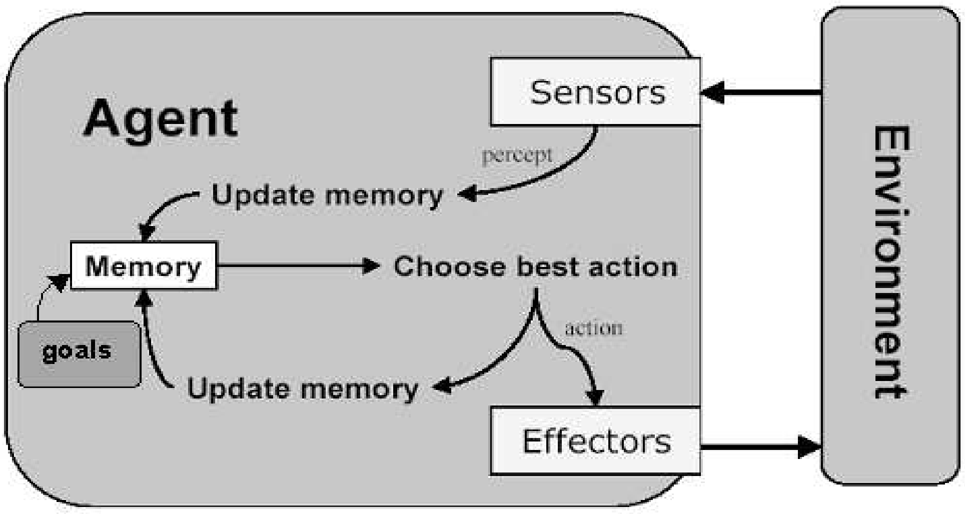

Figure 3-2. Interaction between agent and environment through sensors and effectors in the PAGE paradigm

4 MAS methodology

The MAS methodology is similar to traditional software engineering methodologies [WJK00, D99] but is specialized for use in the distributed agent paradigm. The methodology follows the basic levels shown on the Figure 4-1 actually defining the system itself.

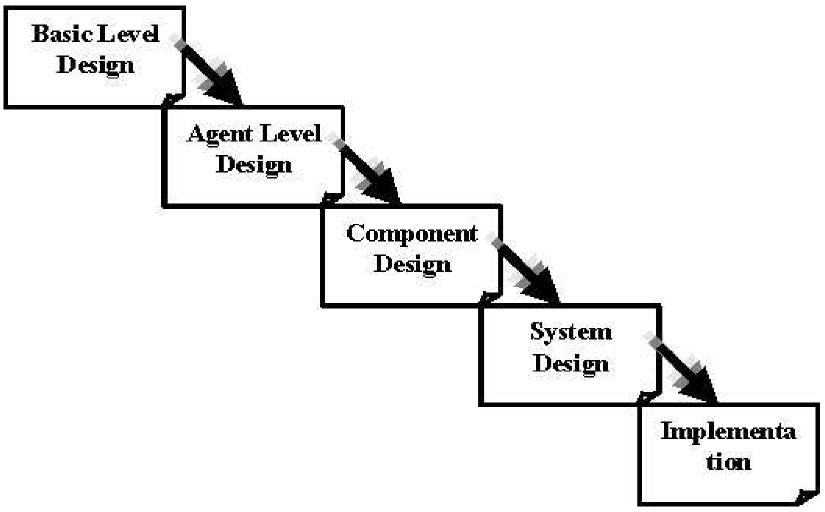

Figure 4-1. MAS methodology

4.1.1 Basic Level Design

The first step in MAS methodology is *basic level design*, which captures the basic types and interactions between agents in the system. At this level, whether or not an agent has intelligence, how that intelligence is captured, or how the agent is defined is not important.

Basic level is concerned with only the high-level definition of the types of agents, their goals, and their external interfaces. It is these external interfaces that define communication protocols between the agents. In the next step how each gent may interact with other agents is defined and which agent types are needed for coordination while leaving the actual numbers, locations, and specific responsibilities of the actual agents within the system to system design phase.

The steps in basic level design are:

1. Identifying agent types and responsibilities.

2. Identifying the possible interactions between agent types.

3. Defining coordination protocols for each type of interaction.

The first step identifies agent types that are analogous to object oriented techniques where the problem space is surveyed for the types and responsibilities of agents needed to effectively model the domain and as it was defined into Chapter 7.2.

Once the types of agents are identified, it is possible to identify kind of interactions that might occur between different types of agents. These interactions become agent conversations that are defined using communication protocols. These protocols describe the possible sequences of messages that may be passed between agents to achieve interaction. On the basic level a possible logical connection between all components of the model is designed and maybe illustrated like in the Figure 4-2.

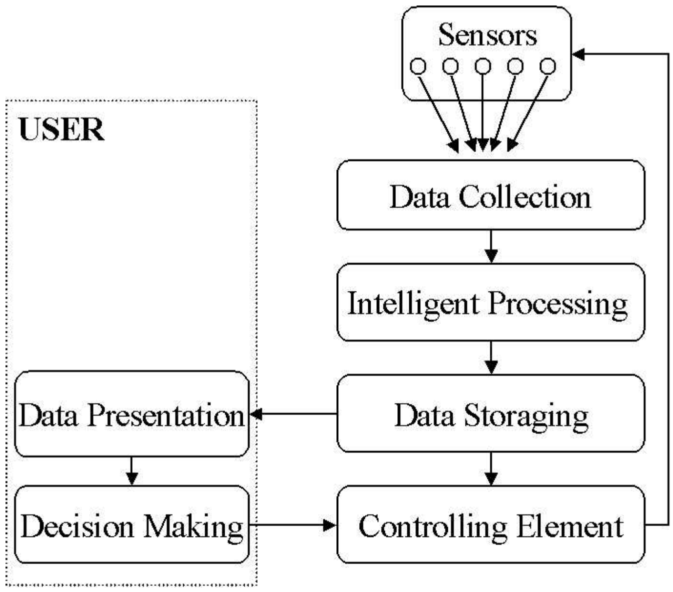

Figure 4-2. Possible logical agent's interactions

4.1.2 Agent Level Design

The next step in MAS development is the *agent level design*. It is at this level that the agent architectures are defined for each individual agent type. The choice of agent architecture is a key issue in the design since it influences other design activities. The selection is based on functional considerations. There are many agent architectures in the literature [WJ95], which were studied prior to starting the design. The architecture used in the system is made up of generic components that may be adaptable for other applications. These generic components might be transformed according to the roles and functionalities of the agent in the organization.

As it was told about agents' cognitive as well reactive capabilities were needed. Cognitive agents would be responsible for management of user interaction and user modeling. Meanwhile, reactive agents would perform extraction of information on the demand. In both, the agent architecture should provide facilities to manage threads of communication independently.

Specific design steps at the agent level are:

1. Mapping actions identified in agent conversations to internal components.
2. Defining data structures identified in agent conversations. These data structures represent input or output from the agent.

4.1.3 Component Design

The *component design* level is the next level of the MAS methodology. Once the agent architecture is defined, the components specified must be designed. However, if components exist that can be re-used, it is our hope that we can define agents in such a way as to take advantage of existing component enabling technologies, such as JavaBeans [JAVABEANS], to allow us to reuse many components. Obvious agent components include:

- planners,
- inference mechanisms,
- searching algorithms,
- learning algorithms and
- analyzing algorithms.

4.1.4 System Design

The next step *system design* takes place once the design of the domain, agents, and components are complete. By defining the domain first, system design becomes an exercise in picking the number and types of agents needed as well as defining specified parameters within the agent definition. Although not technically part of system design, once a system has been defined we can verify certain properties of interest such as safety and liveness. Specific steps in system design include:

1. Selecting the agent types that are needed.
2. Determining the number of agents required of each type and defining:
    a. The agent's physical location or address.
    b. The types of conversations that agents will be able to hold.
    c. Any other parameters defined in the domain.

Defining a domain and system requires a set of formal tools to be able to reuse existing components, synthesize new components, and analyze various properties of the system.

4.1.5 Implementation

Finally after the whole multi-agent model is described the last step remains, *implementation.* On this level a realization takes place using some programming tools and languages. Nowadays these tools usually have to be platform-independent, flexible and in our model provide agent framework so for example Java [JAVASUN] might be applied.

5 Distributed Sensor Networks (DSNs) and Data Fusion

Distributed sensor networks (DSNs) have recently emerged as an important research area [KEW02, HSIGEG01, YGE01]. Advances in sensor technology and computer networking have spurred this development. Even though it is economically feasible to implement DSNs, there are several technical challenges that must be overcome before DSNs can be used for today's increasingly complex information data fusion. Data fusion tries to solve such tasks as battlefield surveillance, remote sensing, global awareness, etc., are usually time-critical, cover a large geographical area, and require reliable delivery of accurate information for their completion.

A lot of models have been proposed for the design of DSNs [CCGK99]. Between them several efficient DSN architectures have been presented in the literature, including the hierarchical and committee organization and tree architecture. While improving the performance of DSNs in different aspects, all these approaches use a common network computing model: the client/server model, which supports many distributed systems, such as remote procedure calling (RPC) [RPC], common object request broker architecture (CORBA) [CORBA], etc. The debate between centralized and decentralized systems is fundamentally about topology – in other words, how the nodes in the system are connected.

Distributed systems often have a more complex organization than any one simple topology. Real-world systems often combine several topologies into one system, making a *hybrid topology* [HDA98]. Nodes typically play multiple roles in such a system. For example, a node might have a centralized interaction with one part of the system, while being part of a hierarchy in another part. A new wave of peer-to-peer system is advancing architecture of centralized systems embedded in decentralized systems, which are presented further in this chapter.

This hybrid topology is realized for example with hundreds of thousands of peers in the file-sharing system used in Gnutella [GNUTELLA], KaZaA[KAZAA], and Morpheus [MORPHEUS]. Most peers have a centralized relationship to a "supernode," forwarding all file queries to this server (much like a Napster [NAPSTER] client sends queries to the Napster server). But instead of supernodes being standalone servers, they band themselves together in a Gnutella-like decentralized network, propagating queries. Internet email also shows this kind of hybrid topology. Mail clients have a centralized relationship with a specific mail server, but mail servers themselves share email in a decentralized fashion.

5.1 General DSN

A general DSN architecture is illustrated on the Figure 5-1, which consists of a set of sensor nodes, a set of processing elements (PEs), and a communication network interconnecting the various PE(s). One or more sensors are associated with one PE. One sensor can report to more than one PE. A PE and its associated sensors are referred to as a cluster. Data are transferred from sensors to their associated PE(s) where the data integration takes place. PE(s) can also coordinate with each other to achieve a better estimation of the environment and report to higher level PE(s). Notice that only the lowest-level PE(s) are connected to the sensor nodes. Higher-level PE(s) only connect to lower-level PE(s), but not the sensor nodes.

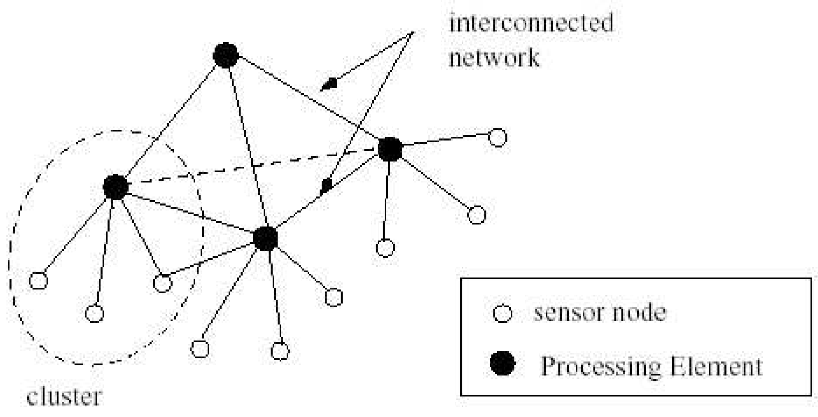

Figure 5-1 The architecture of a general DSN

Figure 5-2. An example of one cluster in DSN and the process of collecting data from sensors is shown on the Figure 5-2.

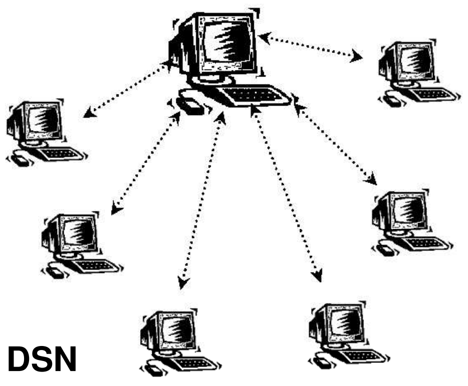

Figure 5-2. An example of one cluster in DSN

In client/server model, the client (individual sensor) sends data to the server (processing element) where data processing tasks are carried out. However, the client/server model is not appropriate for data integration in DSNs:

1. The data integration at the server requires data transfer from local sensor nodes. When the size of data file is large and the number of sensor node is big, the network traffic can be extremely heavy, resulting in poor performance of the system.
2. Suppose connection-oriented service is used (e.g. ftp application uses TCP protocol), the client/server model requires the network connection to be alive and healthy the entire time a data transfer is taking place. If the connection goes down, both the client and the server have to wait until the connection is recovered to finish the data transfer and do further analysis, which will affect the system performance as well.
3. The client/server-based DSN cannot respond to the load changing in real time. When more sensors are deployed, it cannot perform load balancing without changing the structure of the network.

Recent advances in sensor technology allow better, cheaper, and smaller sensors to be used in both military and civilian applications, especially when the environment is harsh, unreliable, or even adversarial. A large number of sensors are usually deployed in order to achieve quality through quantity. On the other hand, sensors typically communicate through wireless networks where the network bandwidth is much lower than for wired communication. These issues bring new challenges to the design of DSNs:

1. data volumes being integrated are much larger,
2. the communication bandwidth for wireless network is much lower and
3. the environment is more unreliable, causing unreliable network connection, noisy background, and increasing the likelihood of input data to be faulty.

5.2 MADSN - An Extension of DSN

There is one extension of DSN that is based on applying mobile agent technologies and called mobile-agent DSN (MADSN) [QIC01]. The difference between them is that MADSN is an improved DSN architecture that uses mobile agents (Figure 5-3). The main idea of it is to make a partly analyzing of data on the low-level peer and then send it to high-level node.

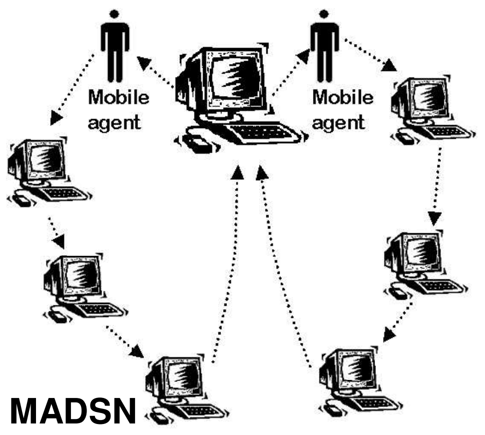

Figure 5-3. An example of one cluster in MADSN

MADSN adopts one more computation paradigm: data stays at the local site, while the integration process (code) is moved to the data sites. The transmitting the computation engine instead of data offers some benefits:

a) Network bandwidth requirement is reduced. Instead of passing large amounts of raw data over the network through several round trips, only the agent with small size is sent. This is especially important for real-time applications and where the communication is through low-bandwidth wireless connections.
b) Better network scalability. The performance of the network is not affected when the number of sensor is increased. Agent architectures that support adaptive network load balancing could do much of a redesign automatically.
c) Extensibility. Mobile agents can be programmed to carry task-adaptive fusion processes, which extend the capability of the system.
d) Stability. Mobile agents can be sent when the network connection is alive and return results when the connection is re-established. Therefore, the performance of MADSN is not much affected by the reliability of the network.

The problem of collecting and analyzing data is the most important topics to be discussed. The process of collecting, analyzing and presenting data to end-user is called ***data fusion***. In the next chapter there is a description of two kind of data fusion, that I have characterized as CDF and DDF.

5.3 Data Fusion in DSNs

5.3.1 Centralized Data Fusion

The architecture of Centralized Data Fusion (CDF) is characterized by a hierarchy of nodes, in which all information is passed up the hierarchy to a centralized fusion node. This sort of architecture has typically been used for information processing.

As might be expected, this architecture usually takes a long time to propagate the fused data back to the lower level nodes. There are several reasons for this. One is related to the character of the data, which are often more complex in DSN and potentially more uncertain than, for example, radar track data. Much of the data have in the past required human analysis to interpret, correlate, and fuse. Attempts to automate this process have proven much more difficult than the automation of processing of data from fire control radars and similar sensors. Another reason is that the primary purpose of the fusion system is to supply a fused picture to the decision-makers at the top of the hierarchy, and so the system need not be optimized for providing a timely fused picture to the lower nodes in the hierarchy.

There are substantial disadvantages to this architecture

1. it imposes a latency on the availability of the fused picture at the lower level nodes,

2. it also imposes a single point of failure in the system, where if the fusion node is lost or loses communications, the overall situation awareness of the system is compromised,

3. it limits the ability of the individual participants to operate independently or as part of a much smaller group and

4. this architecture can require significant communications bandwidth.

5.3.2 Decentralized Data Fusion

Decentralized Data Fusion (DDF) is an evolving technology, concerning the problem of how to fuse data from multiple sensors in order to make a more accurate estimation of the environment. Applications of data fusion cross wide spectrum, including environment monitoring, automatic target detection and tracking, battlefield surveillance, remote sensing, global awareness, etc. They are usually time-critical, cover a large geographical area, and require reliable delivery of accurate information for their completion.

So far, client/server computing model has been most popularly used in Distributed Sensor Networks to handle multi-sensor data fusion. However, as advances in sensor technology and computer networking allow the deployment of large amount of smaller and cheaper sensors, huge volumes of data need to be processed in real-time.

A decentralized data fusion system consists of a network of sensor nodes, each with its own processing facility, which together do not require any central fusion or central communication facility. In such a system, fusion occurs locally at each node on the basis of local observations and the information communicated from neighboring nodes. At no point is there a common place where fusion or global decisions are made.

A decentralized data fusion system is characterized by three constraints:

1. There is no single central fusion center; no one node should be central to the successful operation of the network.

2. There is no common communication facility; nodes cannot broadcast results and communication must be kept on a strictly node-to-node basis.

3. Sensor nodes do not have any global knowledge of sensor network topology; nodes should only know about connections in their own neighborhood.

This architecture has the benefits of guaranteeing each node access to the entire fused picture of the space, incorporating all available information. Since fusion is formed at each node in the hierarchy, the flow of information back down to subordinate nodes is much quicker than in the centralized architecture. Further, the processing burden at higher nodes is reduced because fusion is performed at intermediate stages. The bandwidth requirements are less than if the CDF is applied to this problem. However, there is still a substantial processing requirement associated with maintaining multiple fused pictures.

5.4 Peer-to-Peer

To realize DDF I have used some of the ideas found from Peer-to-Peer systems. Therefore here is a short introduction to some of their concept. A P2P *system* is a collection of distributed resources (desktop and hand-held hosts, devices with embedded processing resources such as digital cameras and phones, or tera-scale supercomputers) connected by a network. They are accessible by others peers directly without passing intermediary entities. The participants of such a network are thus resource (service and content) providers as well as resource requestors (Servent-concept) [S01]. The term "Servent" presents the capability of a node at the same time to be as server well as client: may provide services to others peers and may consume services from others peers. However, our model in more precise is called "Pure" Peer-to-Peer network where any single terminal entity can be removed from the network without having the network suffering any loss of network service.

P2P networks differ from the client-server interactions that typify applications of TCP/IP protocol. A client-server scenario like the WWW [WWW] depends on a single server storing information and distributing it to clients in response to their requests. The information repository remains essentially static, centralized at the server, and subject only to updates by the provider. Users assume a passive role in that they receive, but do not contribute, information. A P2P network, on the other hand, considers all nodes equal in their capacity for sharing information with other network members. Each user makes an information repository available for distribution, which, combined with anyone's ability to join the network, leads to the fast growth of a network composed of distributed information repositories.

P2P networks can use protocols that make it easier for individual nodes to participate and share information, but the trade-off can be decreased security or quality of service. For example, using a lightweight alternative such as the user datagram protocol UDP may sacrifice the reliability that a protocol such as TCP offers [UDPTCP].

A P2P architecture is illustrated on Figure 5-4 that shows an absence of central server and possibility to interact between any participant of the P2P system.

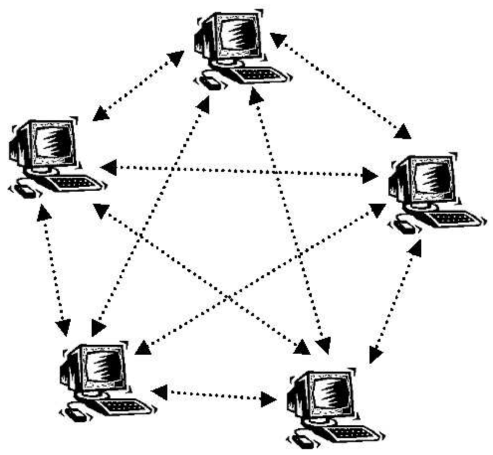

Figure 5-4. P2P Architecture

Distributed application integration that uses software engineered from the bottom up is assisted by Web service standards such as UDDI (Universal, Description, Discovery and Integration) [UDDI], OASIS (Organization for the Advancement of Structured Information Standards) [OASIS] and BPML (Business Process Modelling Language) [BPML] as well as data representation standards such as XML (eXtended Mark-up Language) for example.

Other specifications concerned with interoperability among autonomous systems, such as FIPA (the Foundation for Intelligent Physical Agents) [FIPA], will play a fundamental role in development moving forward. FIPA provides a standardized definition of common semantics for cooperation and coordination, in essence, a lingua franca for defining complex tasks that can be distributed among multiple peers. In addition to defining the nature of interactions, FIPA also provides models for locating services that offer specific functions, such as an intelligent directory service.

Several advantages that make P2P networks attractive are further described in Chapter 9.

6 Peer-to-Peer Distributed Sensor Network Model (P2PDSN)

This chapter presents the Peer-to-Peer Distributed Sensor Network Model (P2PDSN) as was developed as part of the research work. This model was chosen because P2P does not present a binary choice between centralization and decentralization, but represents a way of decentralizing those aspects of a system that can be better handled at the edges of the network. P2PDSN model has proven to be very useful in many applications, e.g. file sharing, anonymous communication, group multicast, load balancing, distributed storage etc.

The model consists of five main objects, from which 4 are different types of agents and the fifth one the user. All these agents together create a distributed sensor network system that can be implemented using Peer-to-Peer technologies. MAS are particularly suited for applications that are to be deployed in highly dynamic environments and subject to incomplete and imprecise information and therefore is an appropriate basis for modeling and building P2P applications.

The P2PDSN model is based on four logical entities: Sensors (O), Controllers (C), Analyzers (A) and Viewers (V). The Figure 6-1 shows a sample model view.

A P2P distributed system based on the model is composed of a self-organized overlay network of interconnected Controllers with the Sensors in their view horizon, topology of which is changing all the time. Each Controller is a peer entity capable of performing computations and hosting resources. Viewer handles requests, which originate from users by means of generating requests to Controllers or to Analyzers. The Controller and Analyzer try to satisfy the request.

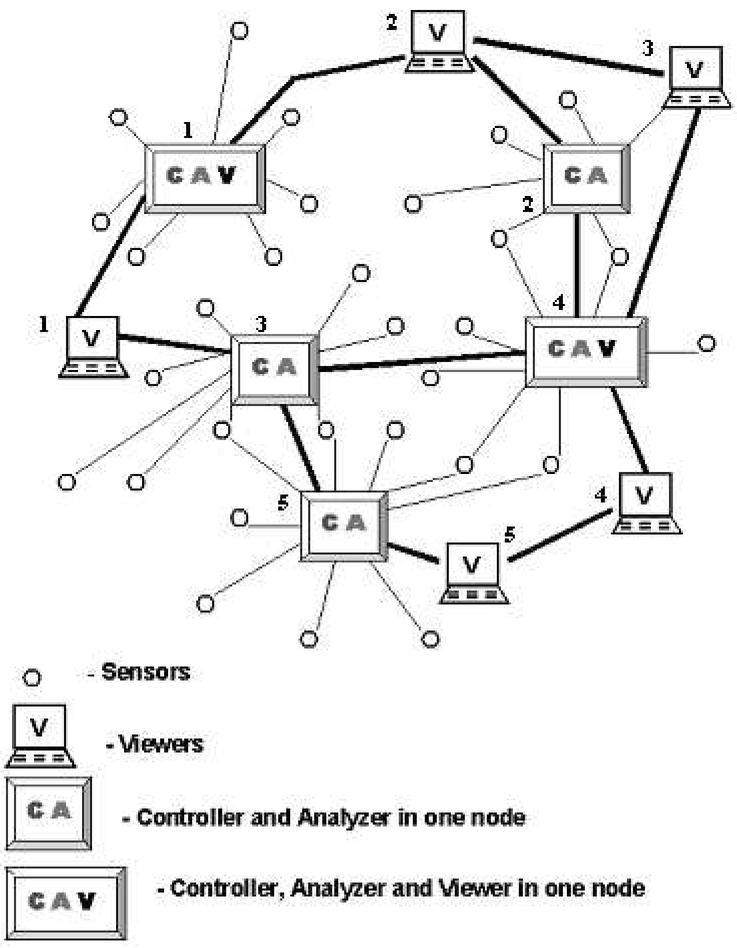
1
CAV
2
V
3
V
CA
2
1
V
3
CA
4
CAV
5
CA
4
V
5
V
- Sensors
- Viewers
- Controller and Analyzer in one node
- Controller, Analyzer and Viewer in one node

Figure 6-1. A sample model view

Controllers interact directly with each other by modifying their environment through information they got from each other and from Sensors that belong to their view horizon. For example, CAV1 (Figure 6-2(a)) has such territory view: {V1, CA3, V2, CA2}. If CAV1 needs other territory view it would start to look for some other CAs that have their own view horizon. So CAV1 would send a flooding request to get data it needs. If CA gets a response (let's suppose it would be CA2 with own view horizon (Figure 6-2(b)), then CAV1 will make a direct connection to CA2. Then depending on a number of hops, which are defined by CA2's horizon view CAV1 will get the new results that is illustrated on Figure 6-2(c). This is a forming of direct communication in P2P systems.

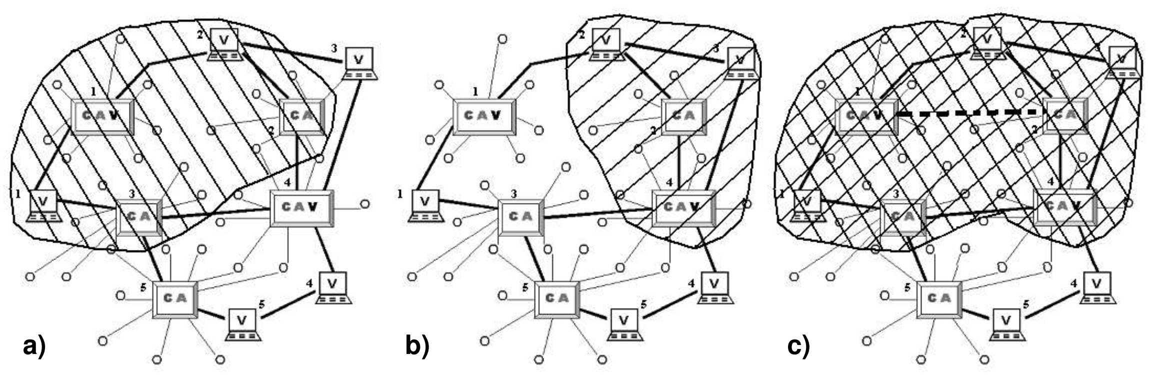

Figure 6-2. View horizon changes

The network is characterized by the absence of a fixed structure, as Sensors also Controllers come and go and discover each other on top of a communication substrate. Each Controller is middleware software capable of performing computations and hosting resources. Each Viewer interacts with local Controller that provide for him a presenting data that was asked. An example of such application may be a GUI system.

The whole model consists of several major functional modules:

- collecting of measurements from sensors,
- analyzing of collected data,
- representation of data to user and
- controlling and monitoring the current state of Distributed Sensor Network.

Controllers handle requests coming from Viewer by generating of collection and analyzing information, interacting with other Controllers, Sensors and Analyzers in order to accomplish their task.

In the thesis MAS is viewed as a further abstraction of the agent-oriented paradigm where agents are at an even higher level of abstraction than typical objects starting from the next chapter till the end of the thesis. Instead of simple objects, with methods that can be invoked by other objects agents coordinate their actions via communication to accomplish individual and community goals. This point of view sidesteps the issues regarding what is or is not an *agent*. Agents are shown as a convenient abstraction, which may or may not possess intelligence. In this way, was handled intelligent and non-intelligent system components equally within the same framework.

7 System Based on P2PDSN model

7.1 Overview

Earlier chapter resented a model for distributed sensor networks. For the model as an environment was chosen a system that is called as Peer-to-Peer Distributed Sensor Network for the role of which was suggested a *complex adaptive system* (CAS). In the CAS framework, the model consists of large numbers of autonomous agents that individually have very simple behavior and that interact with each other in very simple ways. Despite the simplicity of its components, CAS exhibits what is called emergent behavior that sometimes complex and unpredictable.

In the system individual nodes connecting to the network can access a real-time index of other active nodes and of the data they share. In turn, as soon as they connect, these nodes become part of the index themselves, with the data they choose to share automatically added to the index. Because the index provides addresses for resources available at any given time, a member node can simply initiate a direct connection with any connected member node that currently holds the requested information.

Most of P2P network features translate into a rather unstructured, almost primitive information-sharing network that quickly builds itself into a comprehensive information collection. This structure also allows effective sharing of information by large communities at low cost. What renders CAS particularly attractive from a P2P perspective is the fact that the global properties of both systems overlap greatly:

- Decentralized,
- High-available,
- Scalable,
- Robust and
- Fault tolerant.

All these properties are fundamental requirements of the developed model including also these ones:

- Absence of single point of failure,
- Absence of global coordination,
- Extensible and
- Heterogeneous.

Also there are some others features that characterize P2P technology but these are the most describing ones the developed model. For a comprehensive comparison between other approaches review the comparison part of the thesis in Chapter 9.

The next two chapters I will concentrate on defining a system based on the model starting about agents and further going through the whole AOIE paradigm and states of the system.

7.2 Agents in the System (A)

Five interacting parties were chosen in the model:

1. Sensors,
2. Controllers,
3. Analyzers,
4. Viewers and
5. User

From which four of them are agents:

1. Sensor-agent (SA),
2. Controller-agent (CA),
3. Analyzer-agent (AA) and
4. Viewer-agent (VA)

As it was mentioned before each agent can be described by PAGE paradigm. It shows what does the agent has as on input – percepts and based on goals different actions will be executed defining the agents' outputs. Finally the environment defines in what kind of environment the agent lives.

7.2.1 Sensor-agent

Sensor-agent (SA) is located in a place where there is a device, which can measure data from sensor(s). The main target of him is to get measurements from sensors and then send it to other type of agents that are interested in collecting such information. In the Table 1 there is a PAGE-table for Sensor agent illustrated. SA co-operates with Controller agent. The way of their communication will be described in chapter 7.4 Interactions (I).

Agent	**Percept**	**Actions**	**Goals**	**Environment**
Sensor-Agent (SA)	• Sensor's measurement • Control messages from CA • Measurement subscriptions • Sensor discovery messages	• Store measurements • Reply to discovery messages • Send measurement data to CA • Control sensor's properties based on control messages • Maintain a list of CAs with priority	• Maximize the priorities of CAs • Send info about sensor's state to CA • Send data to CA about measurements to CA	Distributed sensor network with sensor measurement indications

Table 1 PAGE-table for Sensor-agent (SA)

7.2.2 Controller-agent

Controller-agent (CA) is located in a place where there is enough bandwidth available for collecting huge amount of data from sensors and other CAs. In the Table 2 there is a PAGE-table for Controller-agent describing what it has as an input and output. CA co-operates with SA, AA and VA. When a new sensor is added to a system CA would find it by itself and then would register to SAs CA is managing a list of all SAs , CAs and VAs that are in its view horizon The way they communicate will be described in chapter 7.4 Interactions (I).

Agent	**Percept**	**Actions**	**Goals**	**Environment**
Controller-Agent(CA)	• Sensor's data • Analyzed data from AA • User's request for analyzed data • User's commands for sensors	• Send commands and requests to sensors • Give measurements to AA • Store analyzed data from AA • Request sensor's measurements from neighbor CA or SA • Present the analyzed data to user • Send commands to neighbor SA and CA	• Collect measurements about sensors data • Control and monitor sensors • Control and monitor the current state of DSN	Distributed sensor network

Table 2. PAGE-table for Controller-agent (CA)

7.2.3 Analyzer-agent

Analyzer-agent (AA) is located in the same place as CA and requires processor resources for executing analyzing algorithms. The main task for the agent is to analyze data, which it gets from CA using special algorithms for analysis. These algorithms may include such as Kalman Filter for predicting trajectory estimates and Transferable Belief Model for identifying the objects. Some of these algorithms have been detailed for example in Sergij Nazarko master's thesis [N02]. The thesis contains a description how these algorithms work and what they can produce as an output. Also an application demonstrating the behavior of these algorithms is presented in thesis. The Table 3 shows a PAGE-table for Analyzer-agent. AA can co-operate only with CA. The way they communicate is described in chapter 7.4 Interactions (I).

Agent	**Percept**	**Actions**	**Goals**	**Environment**
Analyzer-Agent (AA)	• Data from heterogeneous sensors	• Execute analyzing algorithms • Send analyzed data to CA	• Produce analyzed data for CA	Controller-Agent

Table 3. PAGE-table for Analyzer-agent (AA)

7.2.4 Viewer-agent

An agent that carries out the real work for the user is the ***Viewer-agent (VA)***. It interacts with user and can be located with AA and CA but also might be stand alone for example in mobile devices. In the Table 4 a PAGE-table for Analyzer-agent is shown. AA can co-operate with CA, and VA. The way of their communication will be described in chapter 7.4 Interactions (I).

Agent	**Percept**	**Actions**	**Goals**	**Environment**
View-Agent (VA)	• Analysed data from CA • User's request • User's commands	• Request analysed data from CA • Present the analysed data to user • Control sensors via CA	• To fulfil user requests by showing current state of required region (graphically producing the picture) • Control and monitor sensors	Distributed sensor network with user

Table 4. PAGE-table for Viewer-agent (VA)

7.3 Organization of the System (O)

An Organization can be considered from an external point of view, i.e. the Organization is viewed as an external knowledge, which is expressed through the combination of groups whose semantics are different from group of components. The organizational structure can be viewed as decomposition into several levels. There are five main levels as it was mentioned before. The organization of the system is illustrated on the Figure 7-1.

On the Sensor's level it can be different kind of sensors, which measure data and send it on the Controller agent (CA) requests based on the subscription. Controller agent and Analyzer agent have to be all the time together because there is no reason to send a huge amount of data that has not been processed. Also depending on the location of the user Viewer agent either can or cannot. How all these components interact with each other in proximity of Controller agent will be discussed in the next part Interaction as (I) of AOIE paradigm

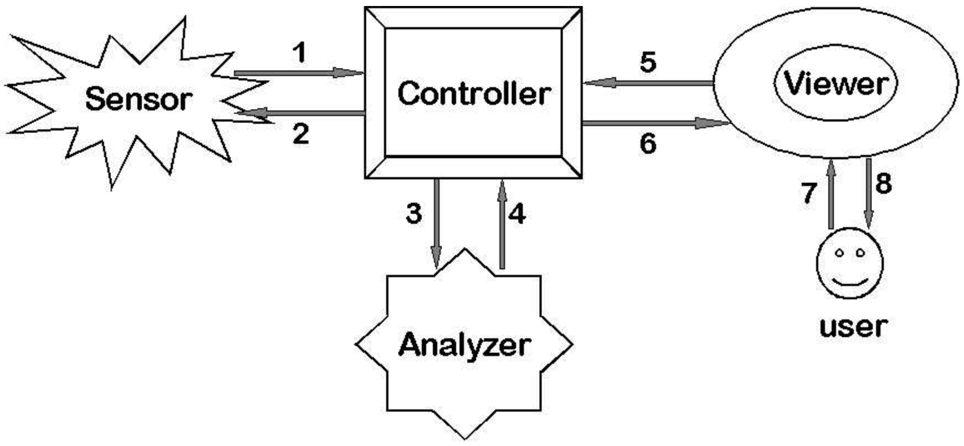

Figure 7-1 Organization of agents and their interactions

As shown in Figure 7-1 relations between all components are the following:

"1" – (CA – SA) Control messages

“2” – (SA – CA) Sensors’ measurements

“3” – (CA – AA) Push measurement data

“4” – (AA – CA) Push analyzed data

“5” – (VA – CA) Request for getting analyzed data, Request for commands

“6” – (CA – VA) Analyzed data

“7” – (user – VA) User’s requests and commands

“8” – (VA – user) Representation of analyzed data

7.4 Interactions (I)

As organization has been defined and the relations shown between agents communication need to be redefined. It can be used to construct the message primitives and content, one of the key points for the interaction and the used communication language.

The communication between agents involves more than just simple messages. In this section, in order to get the feel of how the various agents interact with each other, is presented a description of communication between agents in P2PDSN, answering questions such as “how is the communication specified?”, “tracking of the message; should it be or not?” and “what is the structure of data in the messages?”.

7.4.1 Agent-to-Agent Messages

The model implements negotiation as iterated message passing. A negotiation begins when one agent composes an initial message and sends it to some peers. As the messages spread, some copies will be discarded, others completed and returned.

On the Figure 7-2 is shown an example of process of negotiation between agents in the model.

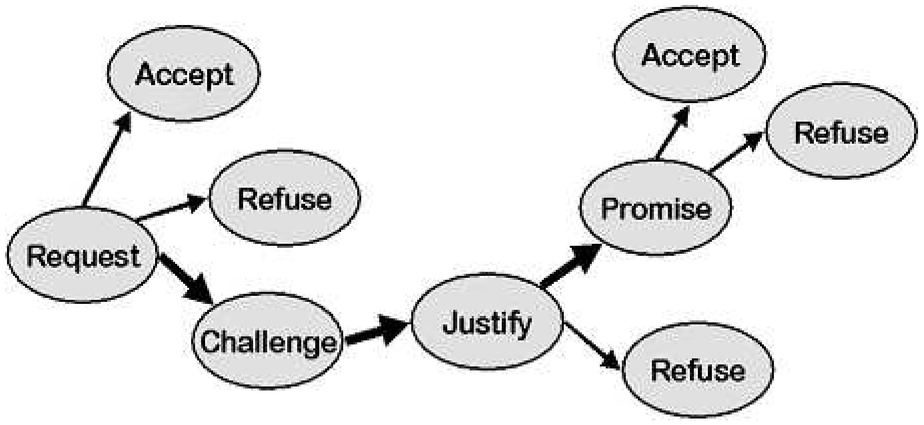

Figure 7-2. Dialog tree

For example in the same way an establishment of connection between agents can be presented.

In the multi-agent system, all agents are searching for updated data. For this purpose they use a communication *messages*. On the example of Viewer agent how it communicates between other Viewers and agents of the system is shown. The message describes the things the agent wants to get or to propose.

Viewer sends a message to CAs or other VAs that it believes can answer its needs. If the agent wants to get new analyzed data from some territory (x1,y1) – (x2,y2), it sends message with such content ("I want to observe territory (x1,y1) – (x2,y2)"). Once Viewer gets a response with the information it needs from CA or VA, they start to communicate directly with one another. They handle the message in a back-and-forth collaboration as each try to complete the transaction.

The basics of inter-agent communications that message structures, and to a lesser extent the protocols for exchange must be formalized and adopted by all agents in the system. Otherwise some conflicts can happen between agents. The model demonstrates the use of formal messages structures and data state of messages in unregulated peer-to-peer environment.

7.4.2 The Structure of Messages

The structure of data carried by a message lets the agent manage the message's distribution to others. State fields include:

- ID of sender,
- recirculation mode,
- expiration time, or time-to-live and
- hop counter.

The *recirculation mode* signals redistribution rules for a message. The *hop counter* can be used by the peer to decide if message is dropped before it expires in case of congestion of the network, while the *expiration time* sets an absolute stop-time. Also this field like the time-to-live field in IP packet should decrement expiration time after each step agent handles the message. If expiration time reached zero, the message has traveled too far and may be discarded.

In the model all agents generally do not track the progress of all the messages they've received or sent. At a high level, an agent needs to know if goals are being achieved, or if a change of strategy is needed. However, because message propagation is directed by independent agents outside any individual's control, a particular message may return in a changed state or may disappear. An agent may attempt to follow the evolution of a single message to get some knowledge about other peers.

7.4.3 Message Paths

Sometimes an agent must wait for information before it can process a message. For example, if a Viewer agent asked for some analyzed data, replies from interested CAs and VAs will arrive only after some delay.

The agent must store the initial "I want to observe territory (x1,y1) – (x2,y2)" message somewhere, so that it may move on to handle other messages in the interim. Rather than keeping a separate list of waiting messages an agent just requeues a pending message to itself. The agent eventually retrieves the recirculating message and tries again to process it. If it fails, it requeues the message yet again. Incoming messages from other traders also simultaneously fall into the queue.

Further on the Figure 7-3 there is an example of communication messages in the model between Controller agent and Sensor agent.

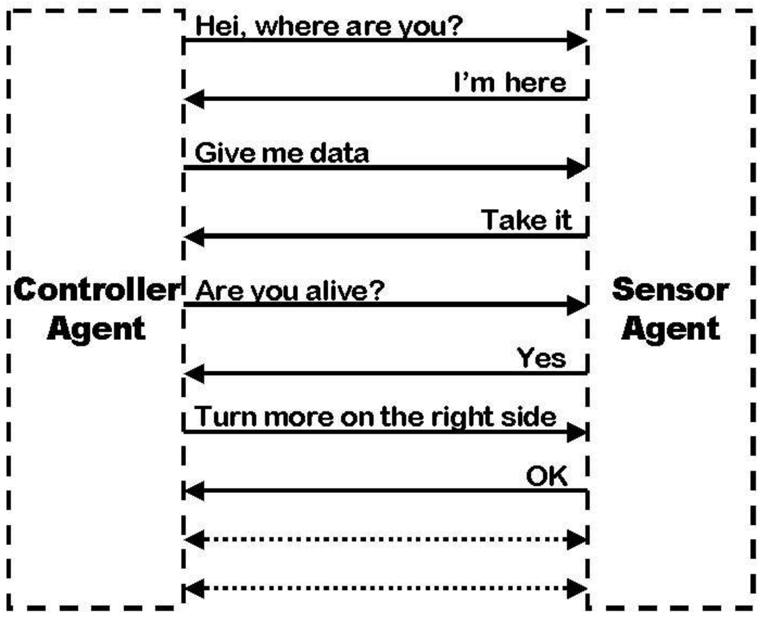

Figure 7-3. Example of sending messages between agents

7.5 Environment (E)

An environment where agents "live" is Distributed Sensor Network. Distribution of DSN's elements depends on the task, which they need to solve. For example, it is possible to use DSN to track quality of output production of whole factory. In this case sensors will be distributed among pipelines and processing elements in different units of the factory. Another example, DSN can be used in robots to collect information from different sensors about weather conditions or tracking flight objects. Sensors can be different devices, which can obtain information from the environment: optic, resistance, radiation, electro magnetic detectors, micro controllers, moisture sensors and etc.

Devices can measure different kind of information in different ways but they cannot obtain data directly from the environment. For these purposes they use communication channels. The simplest way to transfer measured data is to use wired or wireless connection using a transport control protocol TCP, which is stable for errors and failures.

Also more advanced sensor can use wireless networks to transmit data: 801.11b standard or Bluetooth technology. Both technologies are becoming quite popular nowadays. Different kind of devices can support this standard. In the future many kind of home or office devices will support it (fridge, microwave ovens, TV, phone and etc.) to connect into a network.

Using wireless network it is possible for sensors or devices to communicate with each other directly or intermediaries. This is so called ad hoc mode. So devices may exchange data without processing element intrusion.

Agents in the system are programs, which can be executed on the CPU or on any device that supports agent-based platform. As it was mentioned before each agent has sensors that are fixing the current environment state. Through these sensors agents get perceptions that are further analyzed and after making a decision agent returns an action through the effectors, which can change the behavior of device. For example, as in the described model sensors measured some data and collectors request for this data. After analyzing by the controlling element what is the situation in the environment agent controller returns his decision through effectors to sensors what side they have to turn.

A great amount of different agent types can exist in one environment. Each agent has own achievement of the world and what it gets from the environment and what it gives back was described in Chapter 7.2 in PAGE tables. Agent approach becomes more popular, so we can expect in the near future devices with integrated agents. Of course it will be easily possible only if we will have standard for agent communication.

The most important in the environment is users that use services of the network. User can access to the DSN using computer. Also it is possible to use different kind of mobile devices, for example PDA.

In DSN system each component can have a different statements, which are shown in the next chapter.

8 States of the System

Each of the presented agents has their own tasks, responsibilities and goals, which depends on their state. For defining states of the system state diagrams were used. The basic state diagram is shown on Figure 8-1.

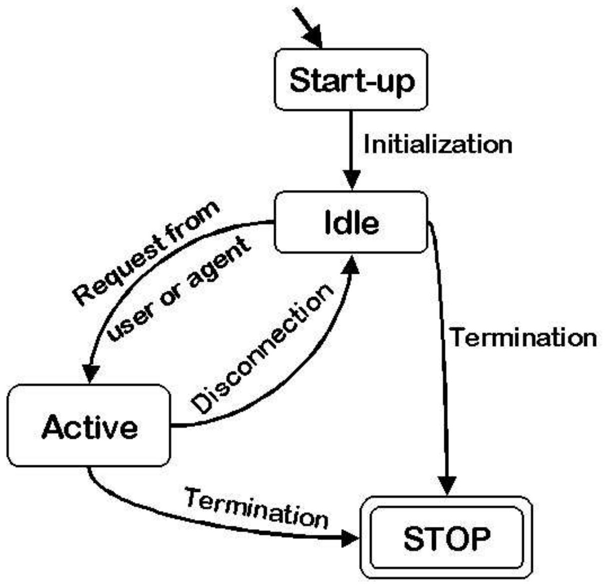

Figure 8-1. An example of state diagram

A state of agent is controlled by the arrival and content of the messages. Further follows a generic states for all agents in the system.

8.1 States for SA

On the Figure 8-2 is illustrated possible states for Sensor-agent. In the beginning it creates a sensor's measurement stream and then comes to the *idle*-state in which it stores the data. Then SA receives a subscription from any of Controller Agents, defines the priority of CA and sends data if the priority is high enough. Also SA can receive command requests from CA for changing of their field of measurement.

Subscription adds new CA to the list of receivers and overload indication message informs that the receiver does not want messages with hop count greater than overload value.

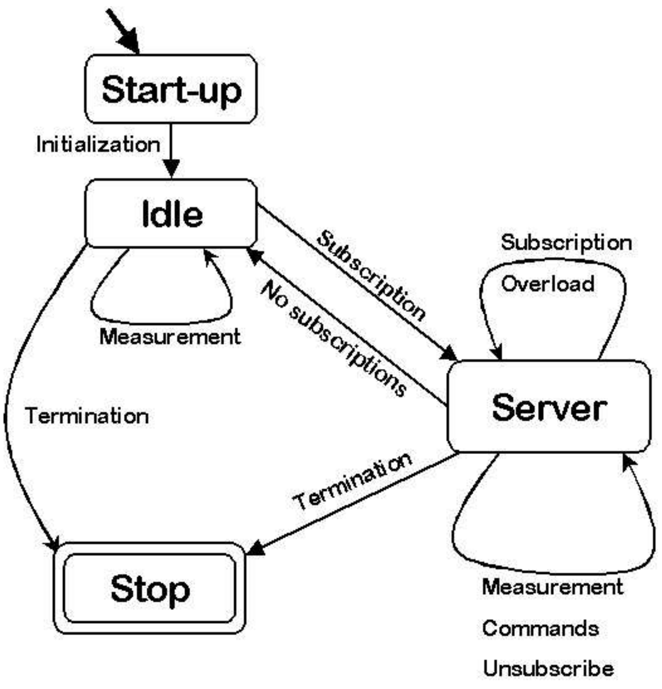

Figure 8-2. SA's state diagram

8.2 States for CA

On Figure 8-3 state diagram for Controller agent is shown. In the start-up state it has to seek out for SAs and other CAs in his own view horizon. Then it sends a permission ID or certificate to SA to make a subscription to sensor measurement streams. After it was accepted CA starts to get measurement data from SAs and pushes them to Analyzer agent. AA automatically gives it back after analyzing. CA also may need information from others CAs, so it sends requests to neighbors to get analyzed data from them. CA can send to SA control messages, which are based on the VA commands and send analyzed data that was asked.

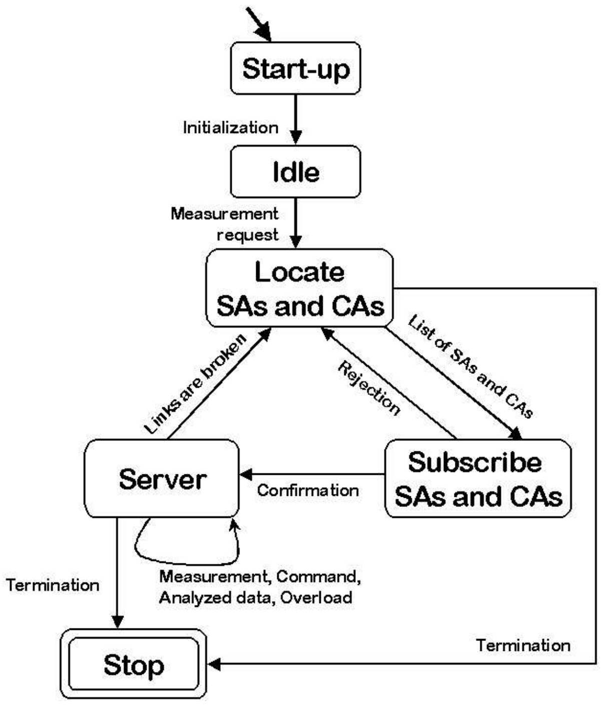

Figure 8-3. CA's state diagram

8.3 States for AA

On Figure 8-4 the state diagram of Analyzer agent is illustrated. It starts to proceed only after CA has pushed a collected data to him.

Then it executes analyzing algorithms, which produce analyzed data as an output. This data will be pushed back to CA. In this data AA can point out to CA which area is good for asking new measurements and also makes a definition of utility of measurements in the way of the optimization of data.

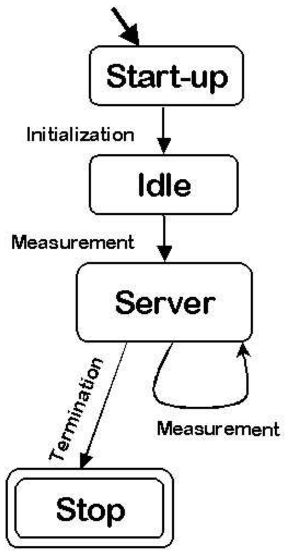

Figure 8-4. AA state diagram

8.4 States for VA

Figure 8-5 presents the state diagram of Viewer agent. It starts to work after user sends requests or commands which he wants to get from the system. VA tries to locate CAs and VAs by sending them its ID or certificate to receive permission on getting analyzed measurements about the region it needs. Then VA “translates” user’s commands and requests into interpretable for other agents primitives/language and sends it to CA.

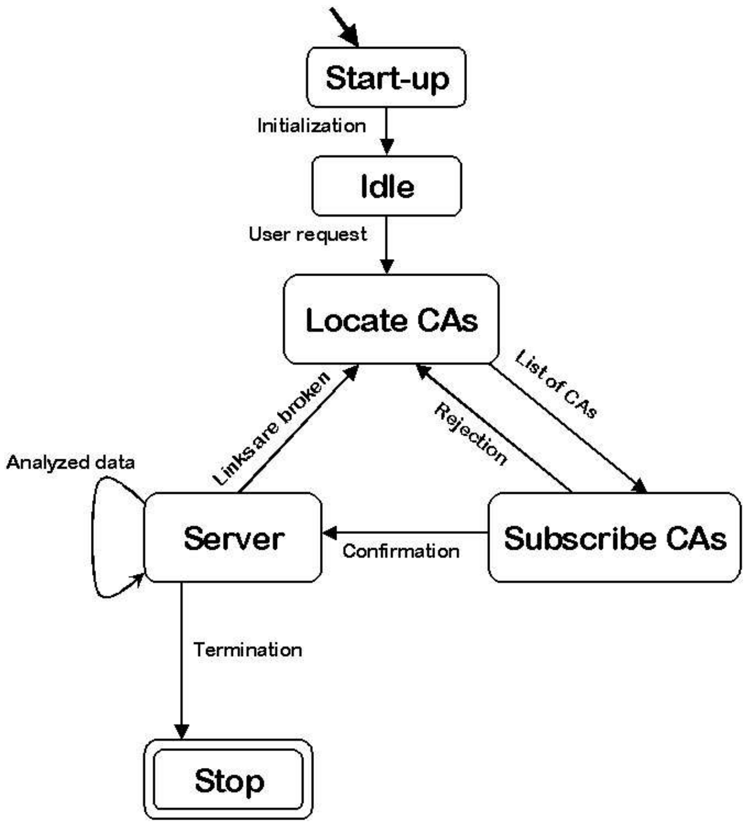

Figure 8-5. VA's state diagram

8.5 Deficiencies in System States

There are four main categories that were not taken into account in the design and require further research:

1. Tampering of agents,
2. Tampering of messages,
3. Unreliability of the network and
4. The implementation of the system

In the model agents are not mobile, but run in a presumably safe environment. Thus, in the thesis the possibility of direct manipulation of an agents' logic by damage of the system is not considered neither the problem of security.

The second risk can be a tampering of messages, which includes *denial of service*, *misdirection* and *loops*. *Misdirection* involves messages specifically chosen to influence an agent's internal decision processes, such as peer selection and decision making algorithms. A message that circulates without ever reaching a conclusion or terminating is *looping*. Looping messages consume resources and can trigger message floods. As with looping packets in TCP/IP networks, looping the systems messages may be difficult to detect, especially if the message goes between two or more states that make it change slightly on every cycle. Message *loops* signify a deadlock or failed negotiation. Agents that route messages incorrectly can also cause loop that is difficult to detect, especially if more than two agents are involved. If not detected, a single looping message can cause an avalanche of traffic resulting in refusal of service.

The third thing left out of the system is a *losing* of messages. It might happen if peers are physically disconnected; message transmission process is interrupted and its sender is not notified about message delivery in unreliable network environment.

As a future work the implementation of the system was left out of evaluation of the system's behavior quantitatively needs to be done in a further research.

9 Comparison between DSN, MADSN and P2PDSN models

As a final step in the research work I made a qualitative comparison between the models illustrated. For doing this I selected the metrics, analyzed the models using these metrics and summarized the results.

Three approaches of existing DSN models: DSN, MADSN and P2PDSN were considered. For analyzing these models some metrics were chosen. It has to be pointed out that there is no ideal model. What kind of model to choose for a system depends on the requirements and the results designers want to get at the end.

In traditional DSN, data is collected by individual sensors, and then transmitted to a higher-level processing element, which performs sensor fusion. During this process, large amounts of data are moved around the network, as is the typical scenario in the client/server paradigm.

MADSN as it was mentioned before is an extension of DSN where mobile agents are used for collecting information and making a partial analysis of measured data.

Peer-to-peer systems approach is the latest addition to distributed sensor systems in which the sharing of resources across the participants can be done in a survivable manner using decentralization. This approach is used by P2PDSN.

9.1 Metrics

For comparing these three models I chose following metrics:

- ***Network topology –*** describes the set of physical or logical links between hosts.
- ***Transferred entities –*** means a type of data that is transferred in a communication channel that can be as wired also wireless.
- ***Bandwidth consumption*** – is a measure of number of bytes of data moved through the communication network in a certain period.

- ***Scalability*** **–** means the system's ability to grow in size. In order to have a scalable system there must be no hotspots (bottlenecks), which control and limit scalability like central servers do.

- ***Extensibility*** – is an ability of system to accept new devices (peers, servers, sensors etc). In DSN context, for example, it is a question how difficult to add one more sensor or processing element to the network. Extensibility depends on the system architecture and restriction.

- ***Load balancing*** **–** is a traffic distribution between a certain amounts of servers.

- ***Information repositories*** **–** are places for storing index data.

- ***Search efficiency*** **–** depends on how well organized the network is, its policies for classifying content and building directories, and the search mechanism itself.

- ***Fault-tolerance*** **–** can be expressed as the ability of a system to sustain traffic flow in the event of a failure. Failures can be encountered in several areas: in the transmission media, in the equipment, and during maintenance. Advanced systems need to be designed to either prevent faults or provide a means to recover from a fault condition and restore traffic without manual intervention.

- ***Information coherence*** – means how efficiently versioning of information can be done. For example multiple copies with many different updates makes information non-coherent, but a centralized replication manager might keep the updates in order making information coherent.

With these metrics defined the characteristic for each of the models could be done.

9.2 Analysis

9.2.1 Network topology

The first metric from which is reasonable to start model comparison is the *network topology* or architecture of them. As was mentioned before that DSN and MADSN have a hierarchical hybrid topology. The difference between them is only in a process of collecting of data.

In DSN sensors belong to a cluster where there is one processing element (PE) that collects sensors' measurements and sends them to central base station (BS). Only BS does a process of analyzing data and managing.

But in MADSN a process of collecting and analyzing of data is a bit different. PE sends mobile agents to collect the data from sensors that are belong to PE territory (or cluster) and by passing through all sensors each mobile agent performs a partial analysis of collected data by using multi-resolution analysis of overlap function that is also used in general DSN but on BS. That means that makes a partial analyzing of measured data is made before transferring analyzed data through the whole network to BS.

P2PDSN architecture of the developed model is completely different from two previous models. There is no any central point. The measured data is collected at exact time from exact sensors by PE request, but any number of PEs can subscribe to the sensors. The cluster of PE can constantly change depending on the view horizon based on link optimization. In this view horizon there can be sensors and also other PEs, which can exchange already analyzed data between each other. Each of PE has its priority that depends on the fitness of the node, (e.g. power of computing machine or maximum bandwidth) so overloading of sensors can be avoided. From the topology point of view this means that everyone can exchange data with every other directly.

On Figure 9-1 the comparison of architectures is illustrated.

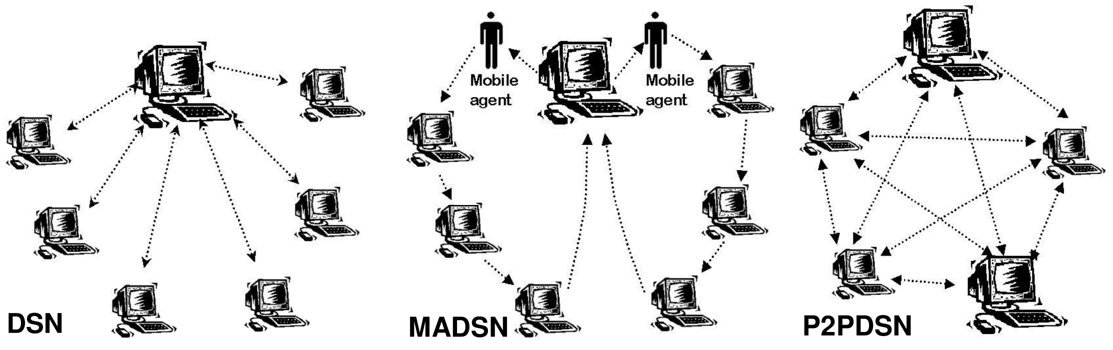

Figure 9-1. Architectures in DSN, MADSN and P2PDSN

9.2.2 Transferred entities

In DSN the measured data is transferred processing elements and then moves to BS. In P2PDSN processed data is transferred also.

MADSN is different because it uses mobile agents and the transferred data is computational.

9.2.3 Bandwidth consumption

In DSN there is not any kind of preprocessing before data is transmitted. Thus sensors transmit all data using connection back end.

In MADSN situation is much better because agents perform analyzing before transmitting data thus decrease amount of traffic dramatically. Of course it's needed to send and receive agent, but the size of them is very small.

P2P bandwidth consumption is in the middle. There are no preprocessing elements in the architecture, but more flexible network architecture allows filtering data quite easily.

9.2.4 Scalability

DSN is not scalable because the performance of PE is always limited and the size of the system proportional to number of PEs. But in MADSN it is much easier to add new sensors to the network where mobile agents can perform a partial analysis on it. So a process of adding new sensors to a cluster insignificantly influences the network load.

P2PDSN is also scalable because there is no central server, which in this case shouldn't synchronize its actions. Only two peers are using the channel through which they are connected. Each component has a view of horizon, which is limited by transmission media, and network load so the view horizon cannot scale without limits.

9.2.5 Extensibility

In DSN and MADSN there is no restriction on the amount of sensors thus one more sensor can be added easily. But it is not so simple with processing elements, if PE is added there might be a connection to existing sensors and this will take some time and make changes on BS.

In P2PDSN all components are extensible, that means sensors and peers can be added easily.

9.2.6 Load balancing

In DSN and MADSN models there is no supporting for load balancing mechanism because several sensors belong to one PE that has a fixed channel for exchanging information. If this exchange overloads the channel bandwidth then this part of DSN will be congested. Also some problems with unexpected technical failure may occur or the central server can be overloaded if there are too many requests for getting data.

But in MADSN mobile agents filter data so a possibility of overloading is much less than in general DSN.

A P2P environment can use a wide range of policies for distributing information. The more refined load balancing techniques monitor traffic and the demand profile for particular information items, then redistribute content to ease the load on individual nodes and possibly locate content closer to points of high demand. Proactive load-balancing schemes can effectively ensure, for example, that a PE does not need to connect to any central point that to get some more analyzed data it needs when the same information resides on a nearby node.

9.2.7 Information repositories

In DSN and MADSN information is stored in hierarchical way on the PEs and BS. So that means it is always known where the information is stored and from which sensor or area exactly it was collected.

In P2PDSN information repositories are dynamic. Nodes can scan each other for desired information and then download it directly from another node. After one of the nodes download information it can make it available for others. Thus, any information with high demand can rapidly spread to many nodes.

9.2.8 Search efficiency

DSN and MADSN use centralized indexing of stored data. It gives some advantages, as it is efficient and fast to find the location of the needed data. Also searches are comprehensive as possible.

In P2PDSN each node has a search engine and an information repository as it was mentioned before. So decentralized indexing of data is used. If one node needs some data it will ask his. After it each node proceeds the search on it's own index and forwards the query if its own index does not match the query. In P2P there are more disadvantages such as slow information discovery and more query traffic on the network.

9.2.9 Fault-tolerance

DSN is not fault-tolerant at all because master nodes always provide single points of failure. If one PE doesn't work there is no other node that can do his work.

In MADSN it is easier to overcome failures because the use of mobile agents provide a possibility to change tasks' collecting paths through sensors. Still in case network becomes partially isolated mobile agents may be blocked up in the partitions. Then a loss of work might become a problem requiring more agents to be sent in the network decreasing the network performance.

In P2PDSN there is no single point of failure. Since in general each node will have multiple peers, data can spontaneously reroute around missing nodes and thus the loss of any single node will only result in the loss of the data resources local to that node. Also this model provides a high availability via automatic replication of data and resolving overload conditions because data is stored into several copies on different computers. That means that instead of just one copy of the data (measurements, analyzed data) there are *n* copies. Then if the primary copy, for example the host it is not available or the disk is crashed, the model allows switching over the copies from other nodes thus generating a new backup. Replicating information at multiple nodes provides a high degree of redundancy, which in turn leads to a high degree of availability and makes it possible to serve more users.

9.2.10 Information coherence

As in DSN and in MADSN all information is stored in one place it is much easier to keep it consistent.

In P2PDSN information is distributed between many nodes so keeping it consistent is more difficult than in DSN and MADSN.

9.3 Summary

The results of the analysis are summarized in Table 9.

Metrics	**DSN**	**MADSN**	**P2PDSN**
Network topology	Hierarchic star	Hierarchic ring	Mesh
Transferred entities	Data	Computation	Data
Bandwidth consumption	High	Low	Medium
Scalability	No	Yes	Yes
Extensibility	Low	Medium	High
Load balancing	No	No	Yes
Information repository	Hierarchic	Hierarchic	Distributed
Search efficiency	High	High	Low
Fault-tolerance	Low	Medium	High
Information coherence	Low	Medium	High

Table 9-1. Comparison between DSN and MADSN from feature points of view

10 Conclusion

The main goal of the thesis was to develop a model that allows disparate systems to interact automatically with one another in order to optimize data fusion and data management processes.

This work has presented the design of a decentralized agent-based peer-to-peer DSN system. As it was described in the thesis multi-agent system suits well for realizing P2PDSN system because of the desired characteristics it possesses which allow agents to represent a peers due to their autonomous and social nature.

From an engineering point of view this has allowed us to provide a system that is *fault-tolerant* because there is no a single point of failure, *high-available,* because there are several copies of the same data but in different locations, *extensible* because new peers (agents) can be added dynamically and *flexible* according to previous benefits. All these properties allow such system to be long lived, without requiring it to be re-compiled or re-deployed frequently.

There wasn't enough time to make a full analysis of the system so still some future work is left that has to be done. The questions yet unanswered are the security, loosing of messages, loops, and denial of services and technical possibilities of devices that can be used in the system. Also the system has to be implemented for more detail testing and verifying that all the components work together.

REFERENCES

[AD98] Armstrong A., Durfee E., “Mixing and Memory: Emergent Cooperation in an Information Marketplace”. In ICMAS 98, 3rd International Conference on Multi-Agent Systems. Paris, France, 1998.

[ATL] Advanced Technology Laboratories homepage, http://www.atl.external.lmco.com/overview.html

[B01] Ballinger, K. “Web Services Interoperability and SOAP”. Microsoft Developer Network Library, May 1, 2001.

[BM01] Frances Brazier, Maarten van Steen, “Multi-Agent-Systems and Applications”, London, 2001

[BPML] BPML home page - http://www.bpml.org/

[BS01] A. Boulis and B. Srivastava, “A Framework for Efficient and Programmable Sensor Networks”, Networked and Embedded Systems Lab, EE Department, University of California at Los Angeles, 2001

[BW98] Brauer W., Weiß G. "Multi-Machine Scheduling - A Multi-Agent Learning Approach". In ICMAS 98, 3rd International Conference on Multi-Agent Systems, Paris, France, 1998.

[CCGK99] A. Chandrakasan, R. Amirtharajah, S. Cho, J. Goodman, G. Konduri. "Design considerations for distributed sensor systems." Proc. IEEE 1999 Custom Integrated Circuits Conference (CICC '99) (May 1999), pp. 279-286.

[CFP] Cheese Factory-Project, December 2002, http://tisu.it.jyu.fi/cheesefactory/

[CDK01] George Coulouris, Jean Dollimore and Tim Kindberg, "Distributed Systems: Concepts and Design", 2001

[CGS98] Claire Tomlin, George Pappas, and Shankar Sastry "Conflict Resolution in Air Traffic Management: A study in Multi-Agent Hybrid Systems", IEEE Transactions on Automatic Control, volume 43(4):509-521, April, 1998

[CORBA] CORBA - http://www.sei.cmu.edu/str/descriptions/corba.html

[D99]	Scott DeLoach, “Multiagent Systems Engineering: A Methodology and Language for Designing Agent System”, AOIS’99, 1999
[D95]	Demazeau, Y., “Interactions to Collective Behavior in Agent-Based Systems”, 1st European Conference on Cognitive Sciences, St. Malo, France, 1995.
[D97]	Demazeau, Y., Steps towards “Multi-Agent Oriented Programming”, 1st International Workshop on Multi-Agent Systems, IWMAS 97, Boston, 1997. (slides)
[D98]	Demazeau, Y., “Preface to Multi-Agent Systems”, ICMAS 98, 3rd International Conference on Multi-Agent Systems, Paris, 1998.
[DARPA]	DARPA home page - http://www.darpa.mil/
[DF98]	Demazeau, Y. & Ferber, J., “Introduction to Multi-Agent Systems”, 3rd International Conference on Multi-Agent Systems, Paris, 1998, (slides) (tutorial)
[FG98]	Ferber J., Gutknecht O., “A Meta-Model for the Analysis and Design of Organizations in Multi-Agent Systems”, ICMAS 98, 3rd International Conference on Multi-Agent Systems, Paris, France, 1998.

[FG99]	Franklin, S., Graesser, A., "Is It an Agent, or Just a Program?: A Taxonomy for Autonomous Agents", Lecture Notes in Artificial Intelligence, Springer, 1999.
[FIPA]	Foundation for Intelligent Physical Agents, “FIPA Home Page” http://www.fipa.org/
[FLM97]	Finin, T., Labrou, Mayfield J., “KQML as an Agent Communication Language, in Bradshaw J”, Software Agents, MIT Press Cambridge, 1997.
[GF92]	Genesereth, M., Fikes, R., “Knowledge interchange format version 3.0 reference manual”, Stanford University technical report: Stanford, CA., 1992
[GGRG98]	Gimenez E., Godo L., Rodriguez-Aguilar J., Garcia-Calves P. “Designing Bidding Strategies for Trading Agents in Electronic Auctions”. ICMAS 98, 3rd International Conference on Multi-Agent Systems, Paris, France, 1998.
[GK94]	Genesereth, M. R., and Ketchpel, S. P., “Software Agents. Communications of the ACM” 37(7): 48–53, 1994

[GM93]	Ginsberg, Matthew L., “The Knowledge Interchange Format: The KIF of Death.” Morgan Kaufmann,. ISBN 1-55860-221-6, 1993
[GNUTELLA]	Gnutella homepage - http://www.gnutella.com/
[GS96]	Garrido, L., and Sycara, K., “Multiagent Systems”, Second International Conference on Multiagent Systems, AAAI Press, 1996
[GT]	Globus Toolkit home page - http://www.globus.org/
[H95]	Hayes-Roth, B., "An Architecture for Adaptive Intelligent Systems", Artificial Intelligence: Special Issue on Agents and Interactivity, 1995.
[HDA98]	L. Hiuchy, M. Dobrucky, J. Astalos, ”Hybrid Approach to Task Allocation in Distributed Systems”, 1998.
[HS97]	Huhns, M.N., Singh, M.P. (eds), "Readings in Agents", Morgan Kaufmann Pub., 1997.
[HSIGEG01]	J. Heidemann, F. Silva, Ch. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan., "Building Efficient Wireless Sensor Networks with Low-Level Naming", Symposium on Operating Systems Principles, Canada, ACM, October, 2001.

[INPG98]	Generalization modeling using an agent paradigm, AGENT-project Technical Annex, 1998
[IUI98]	Itoh F., Ueda T., Ikeda Y., “Example-Based Frame Mapping for Heterogeneous Information Agents, ICMAS 98, 3rd International Conference on Multi-Agent Systems, Paris, France, 1998.
[IW02]	S. Sitharama Iyengar, Qishi Wu, “Computational Aspects of Distributed Sensor Network, Louisiana State University”, EEEI, ISPAN '02, 2002
[J00]	S. Jamenson, “Architecture for Distributed Information Fusion To Support Situation Awareness on the Digital Battlefield”, Artificial Intelligence Laboratory, Camden, USA, 2000.
[JAVABEANS]	Enterprise JavaBeans™ Tutorial http://developer.java.sun.com/developer/onlineTraining/Beans/EJBTutorial/
[JNFOO00]	Jennings, N.R., Norman, T.J., Faratin, P., O'Brien, P., Odgers, B., "Autonomous Agents for Business Process Management", International Journal of Applied Artificial Intelligence, 2000.

[JAVASUN] Java Sun homepage - http://java.sun.com/

[KAZAA] KaZaA homepage - http://www.kazaa.com/us/index.php

[MB98] Malville E., Bourdon T. “Task Allocation: A Group Self Design Approach”, ICMAS 98, 3rd International Conference on Multi-Agent Systems, Paris, France, 1998.

[MORPHEUS] Morpheus homepage - http://www.morpheus.com/

[MUFASION] Multi-Resolution Data Fusion using Agent-Bearing Sensors in Hierarchically-Organized Sensor Networks (MU-FASHION project), http://www.ee.duke.edu/~vishnus/DARPA/darpa.htm

[N02] Sergiy Nazarko, “Evaluation of Data Fusion Methods Using Kalman Filtering and Transferable Belief Model”, Master Thesis, University of Jyväskylä, 2002

[NAPSTER] Napster homepage - http://www.dovebid.com/napster

[NFF91] Neches, R., Fikes, R., Finin, T., Gruber, R., Patil, R., Senator, T. and Swartout, W., “Enabling Technology for Knowledge Sharing. AI Magazine”, Fall 1991. 12(3): pp. 36-56.

[OASIS] OASIS home page - http://www.oasis-open.org/

[PC98] Picault S., Collinot A. “Designing Social Cognition Models for MAS through Simulating Primate Societies”, ICMAS 98, 3rd International Conference on Multi-Agent Systems, Paris, France, 1998.

[PIKM99] L. Prasad, S. S. Iyengar, R. L. Kashyap, and R. N. Madan, “Functional characterization of sensor integration in distributed sensor networks”, IEEE Trans. Syst., Man, Cybern., SMC-21, Sept./Oct. 1999.

[QIC01] H. Qi, S. S. Iyengar and K. Chakrabarty, “Distributed multiresolution data integration using mobile agents”, accepted for publication in Proc. IEEE Aerospace Conference, 2001

[RPC] Remote Procedure Call Tutorial - http://www.cs.cf.ac.uk/Dave/C/node33.html

[RN95] Stuart Russel, Peter Norvig “Artificial intelligence: A Modern Approach part2”, Intelligent agents, 1995

[S97] Y. Shoham, “An Overview of Agent-Oriented Programming”, Software Agents, AAAI, USA, 1997

[S01] R. Schollmeier, “A Definition of Peer-to-Peer Networking for the Classification of P2P Architectures and Applications”, IEEE, 2001

[SD01] J. Sichman and Y. Demazeau., “On Social Reasoning in Multi-Agent Systems”, n°13, pp. 68-84, AEPIA, Madrid, Spain, Summer, 2001

[STX00] F.Sadri, F. Toni, I.Xanthakos, “A Logic-Agent based System for Semantic Integration”, 17th International CODATA Conference “Data and Information for the Coming Knowledge Millennium”, 2000

[UDPTCP] Creative IT, products support UDP and TCP protocols in P2P environment - http://www.creative-it.net/about_en.htm

[UDDI] UDDI home page - http://www.uddi.org/

[VD99] Van Aeken, F. & Demazeau, Y., “Minimal Multi-Agent Systems”, ICMAS 98, 3rd International Conference on Multi-Agent Systems, Paris, France, 1999.

[W01] Wolter, R. “Overview of SOAP Client in Windows XP”, Microsoft Developer Network Library, May 2001.

[WJ95]	Wooldridge, M., Jennings, N., “Agent Theorieties-Oriented Analysis and Design”, Agents’99, Seattle WA, May 1999.
[WJK00]	Wooldridge, M., Jennings, N., & Kinny, D.,“A Methodology for Agent-Oriented Analysis and Design”, Agents’00, Netherlands, 2000.
[WP99]	T. White, B. Pagurek. “Emergent Behavior and Mobile Agents”. In Proc. of the Workshop on Mobile Agents in the Context of Competition and Cooperation at1 Autonomous Agents, 1999.
[WWW]	World Wide Web homepage - http://www.org/
[Y01]	Youll James, “Peer to Peer Transactions in Agent-mediated Electronic Commerce”, Master thesis, Massachusetts Institute of Technology, 2001
[YGE01]	Yan Yu, R. Govindan and D. Estrin., "Geographical and Energy Aware Routing: A Recursive Data Dissemination Protocol for Wireless Sensor Networks" , UCLA Computer Science Department Technical Report UCLA/CSD-TR-01-0023, May 2001