Aspects of cooperation in a Distributed Problem Solving environment

Arne Nylund

Department of Computer Science
Institute of Mathematical and Physical Sciences
University of Tromsø
Box 953
9001 TROMSØ
NORWAY
E-mail: arneny@sfd.uit.no

Abstract

A great deal of our daily activity involves some form of cooperation. This paper identifies general problem solving situations and show their applicability as metaphors for computer problem solving. Scientific communities and organizational theory are examples of such metaphors. Distributed Problem Solving is discussed, and main aspects such as coordination of agents, communication among agents, local agent sophistication, uncertainty and coherence are addressed.

Introduction

Human problem solving has certain characteristics. Various persons usually have different *skills*, such as planning, sensing, the ability to communicate etc. Persons domain knowledge is usually also different. This implies that the problem solvers often have different *appropriateness* for a given problem. These characteristics are also valid for problem solving in a distributed environment. In addition, the problem solvers as one group, must take care of the decomposed sub-problems. They must share their knowledge, and they must share the available resources. Aspects of distributed problems which may be independently distributed include actions, knowledge, resources and work [Bond 88].

Distributed Problem Solving (DPS) can be defined as [Durfee 88](Page vii):

> *"loosely-coupled, distributed networks of semi-autonomous problem solving agents that perform sophisticated problem solving and cooperatively interact to solve problems"*

Loosely-coupled means that each agent spends more of its time computing instead of communicating. In a *loosely-coupled* system the shared computing services are provided by some of the computers in the network and accessed by software that runs in all the computers. A *distributed network* is a collection of two or more computers (or processors) which are interconnected by some physical medium. *Semi-autonomous* means that each agent in the system governs itself to a high degree.

Distribution may be defined in several ways. Bond and Gasser [Bond 88] defines it in term of conceptual "distance" between the distributed elements. According to them this distance can be:

- **Computation cost**; Distribution may be defined in terms of the cost of using knowledge or a special skill because it may vary depending on the location.
- **Spatial distance**; The spatial locations of agents, knowledge and sensor input define an axis of distribution.
- **Temporal distance**; Knowledge and events may appear or become inaccessible through time and may therefore be distributed. Storage is one way of making such knowledge accessible. Whether it can be stored or must be deduced will affect temporal distribution and the utility of the knowledge.
- **Logical or Deductive distance**; The need to rely on intermediary processes to make some new knowledge accessible is a basis for distribution.
- **Semantic distance**; Distribution may be justified by the desire for clustering of knowledge items and their access and use.

An agent in a DPS system can modify its behavior if needed, and it can plan its interaction with other agents. In some sense these agents are intelligent if we define intelligence as the ability to react to changing situations where decisions are based on knowledge. None of the agents in the system has enough information to solve the overall problem. This may be true for several reasons; it may not be possible due to distribution of information or it may not be feasible to distribute the information due to capacity and efficiency considerations. The group (the collection of agents working on the same problem) hopefully has enough information, and it must be shared to allow the group as a whole to produce an answer.

Problem solving may be more efficient if agents explore different parts of the search space concurrently. Sharing partial solutions may enable agents to form results that they previous could not do, or narrow the agent's search space. Thus, effective problem solving must have cooperation [DurfeeB 87]. Cooperation gives the distributed approach much of its power. *However, despite its prevalence, cooperation is still a poorly understood phenomenon within computer science* [Durfee 88].

Metaphors for distributed problem solving

Metaphors are frequently used to suggest a analogy between one kind of object or idea, and another. *The ship ploughs the sea* is an example. Metaphors have also been used within computer science. The most famous one is probably Macintosh's desktop metaphor. It is used to increase the abstraction level of the user interface so that novice users can see the analogy to their working desk with documents, the trash can, etc.

There are several reasons why people cooperate; i.e. to help somebody else, for their mutual benefit, etc. A major reason is probably *self-interest.* We have to admit that most of us are egoistic, so both individuals and groups tend to cooperate if they think they can achieve more by working together than by working alone. Companies cooperate to increase profits and engineers cooperate to solve problems.

Another reason for cooperation may be the complexity of the problem at hand. Both the information a person can absorb and the detail of control he can handle is limited. Simon calls this *bounded rationality* and defines it as [Simon 57]:

> *"The capacity of the human mind for formulating and solving complex problems is very small compared with the size of the problems whose solution is required for objectively rational behavior in the real world - or even for a reasonable approximation to such objective rationality."*

A problem solving agent executing on a processor clearly exhibits the same limitations; a processor can execute a limited number of instructions per second, and this limit both the amount of information it can process and the amount of control it can carry out. These shared limitations indicate that parts of human problem solving situations may be applicable as metaphors for DPS.

As stated earlier, both humans and computers "suffer" from *bounded rationality*. This is a prime factor in the evolution of multiperson organizations, whether it is an unregimented group or more structured alternatives [Fox 81]. By definition an organization is defined as a composition of structure and control regime [Fox 81]. A structure can range from hierarchies to heterarchies, while the control regime range from people being controlled by others to markets where services are negotiated. Organizational theory may therefore provide a metaphor that can be used to coordinate problem solvers so that they cooperate in a more coherent manner. Within organizational theory individual members try to form an overall team. The purpose is to define how each person in the organization is going to act, and the form of interaction with other persons within the same organization. At a high level of abstraction both human organizations and DPS systems addresses the same kind of problems; which agent does what, which resources should be available to which agents and the flow of control and information. These persons' activities and who they are going to cooperate with are described in the organizational structure. Thus, such a structure will help agents in a DPS system decide where to get the wanted information, basically how to manipulate the information and whom to ask for advice, and where to send the results.

This metaphor is suitable when agents have well defined relationships and that this relationship is relatively static. The solution method must also be known i.e. the responsibilities of the different agents. Agents do not have to be very independent, thus, less sophisticated behavior needed.

Scientific communities can also be used as metaphors for studying distribution and cooperation in problem solving processes [Kornfeld 81]. They are very successful at generating and deciding between alternative explanations of phenomena. If we look at the progress within scientific communities there are two different aspects:

- Globally and over many decades the progress seems coherent and purposeful.
- Locally the situation is quite different. At a given moment of time there usually exist many conflicting theories about the same phenomena. Scientists are arguing about these theories. A theory that in the end turns out to be wrong might gain temporary popularity.

These communities are highly parallel systems. Scientists work concurrently on problems all over the world. The problems may be the same, they may be related or they might be quite different. In many cases the scientists don't even know about each other. One thing most of them do is communicating their ideas, goals or results. This might either be done by selecting some partners to cooperate with or like a broadcast to everybody who is interested. Publishing articles or books is like writing on a public blackboard. This communication might have different levels of influence on the work of the others.

Scientific communities exhibit many characteristics which seem useful and motivating when designing DPS systems [Kornfeld 81]:

- scientific communities are concerned with resource control. Research programs which seem promising often get more support than less promising alternatives.
- once a paper has been published, it can never be "erased". The scientist may publish a contradictory article, but the original will always be available.
- scientists can work concurrently without affecting each other or they may affect each other.
- scientists can work concurrently on the same, similar or quite different problems.

A scientific community consists of scientists (problem solving agents) who do research most of the time (loosely-coupled and semi-autonomous). These scientists distribute their work by publishing articles, using E-mail, using telephone, etc. (They are part of a sophisticated form of distributed network). The reason for the overall success within scientific community research is due to its tolerance of its diversity [Kornfeld 81].

Coordination of the scientist's activities is not straightforward. As mentioned before, at a given moment of time the activities of the different scientists do not necessarily look coherent. The reason for this is usually that the scientists view problems differently, they use different paths to get to the wanted solution or they disagree on what the solution is. For their work to be coherent at a given time, the scientists must have knowledge about both earlier work, current work and the intentions of other scientists working on the subject. This requires intelligent behavior of the scientists; they must be able to know where to find previous work, they must be able to plan their own activities, etc.

Scientists seems to be "connected" in a loosely-coupled network. Cooperation between the agents is a result of reflections and ideas of the individual agents. The coordination mechanisms of these communities will therefore rely on highly sophisticated problem solvers.

This metaphor is suitable for problems where agents needs to be independent, where the solution method is unknown, and different alternatives must be developed in parallel. It is also useful when the relationship between the agents changes relatively often.

The scientific community and organizational theory are two different metaphors for designing DPS systems. They cover both edges of the problem space,from highly independent agents to agents with more dependent relationships. For problems between these extremes, it may be useful to use a combination of the two metaphors.

Goals of cooperation

Cooperation is what gives DPS systems their potential power. However, this is not easy to achieve when subproblems are interdependent. The main concern within the field is:

How to get agents who are perfectly willing to cooperate to act as a team?

Cooperation between agents is necessary in a distributed problem solving environment. For any entities to cooperate in an efficient manner, their activities need to be coordinated, thus coordination is an essential part of cooperation. The goals of cooperation can be divided in two; from the network's point of view and from the individual agent's point of view. These do not have to be compatible, and the local viewpoint can vary from agent to agent. From the individual agent's point of view, the goal is to maximize its local performance [DurfeeA 87]. Examples of local goals may be:

- to allocate the resources needed
- to try to verify local hypothesis

Agents who are driven by self-interest must recognize that cooperation is in their own interest. To realize this, they need local-knowledge to guide them into cooperation [DurfeeA 87]. If the problem solvers use goals to guide their decisions, they can cooperate by sharing high-level goals. However, this is not straight forward. If the problem solvers have different views on how to achieve these goals, their different perspectives might lead them towards competing solutions [DurfeeA 87]. Both the labour-party and the conservatives have the same high-level goal; to reduce unemployment. But they have different ideas on how to achieve this goal.

From a network point of view, the goal is to achieve the best possible network performance [Durfee 88]; to increase the task completion rate by the use of parallelism, to increase the scope of tasks which can be handled by sharing resources (physical devices, information and expertise), to increase the probability of completing a given task and to decrease the number of harmful interactions among tasks. In a problem solving environment these generic goals might be transformed into a more specific set of goals [Durfee 88]:

- to increase problem solving speed by decomposition and by solving sub-problems in parallel.
- to minimize agents idle time.
- to improve overall problem solving by allowing agents to exchange information.
- to increase reliability by replication.
- to improve performance by load balancing.
- to reduce the amount of unnecessary duplication.
- to increase confidence in a solution by verifying solutions.
- to develop a wide variety of solutions.
- to reduce the use of communication resources.

All these specific goals are general for a problem solving environment. In a given situation it might be impossible or not even desirable to achieve them all. If speed is vital, time should not be spent on verification of results or developing a wide variety of solutions.

Agents may cooperate in two ways. If their individual goals are compatible, the agents may cooperate unwittingly as they pursue these goals on their own, or they may cooperate intentionally. Agents may also be "forced" to cooperate by having predefined roles. The rest of this paper will emphasis the latter case.

Interaction among agents

Interaction is defined as the collective action in a DPS system which is initiated by one agent taking an action or making a decision based upon the presence or knowledge of another agent [Bond 88]. Interaction is inherently distributed, and it involves at least two agents. Interaction between agents is essential in a DPS system because this is what makes it possible for several agents to combine their efforts in order to solve their common goal.

There are several ways that agents can coordinate each other's activities [Durfee 88]. They can use one agent as a *controller* and send all information to this agent. The controller is then responsible for the coordination. The agents can also broadcast their information, leaving the individual agents to see how cooperation can be done. Or pairs of agents can exchange information and form contracts with each other.

Dimensions of multiagent interaction

When viewing groups of agents, several dimensions of interaction are important. These include [Bond 88]:

- **among whom** does it take place
- **when** does it occur. This is important because knowledge among agents may have temporal relationships.
- **the content** of the interaction, i.e. what results should be shared, which information to acquire, etc.
- **how is it accomplished**, i.e. which agents are involved.
- **why** does it take place, i.e. what triggered the interaction.
- **the basis of commonality**, .i.e. is there a common language, shared context, etc.

In a DPS environment we need to have a language for interaction between the agents. In order to design such a language we need to know what kind of knowledge to represent for communication. In general, agents will have disparate knowledge [Bond 88]. This implies that the language must be able to handle differences in knowledge i.e. communication must succeed in spite of differences in knowledge. It must also allow communication about disparities in knowledge. An example here is two agents who refers the same object, using different terms and having different viewpoints.

The Contract Net Protocol [Smith 80] is an example of a protocol that uses a fixed interaction language [Bond 88].The internode language consists of structured message types; task announcements, bids and awards. These messages were tailored to the types of information needed to solve the Contract Net's primary problem, marketlike task allocation.

Interaction structures

One of the major problems within DPS is to decide which control regime to use. Some research has been done in viewing distributed systems as being analogous to human organizations [Fox 81]. A main conclusion is that many of the same parameters are present within both areas.

An *organization* can provide a framework of constraints and expectations for agent behavior, which focuses on decision making and action [Bond 88]. Organizations found in literature are:

- **Hierarchical organization;** an hierarchical organization is well suited when control or results must be concentrated at one point. Agents with more global information

guide agents with less global information. Decision-making data flow upward in progressively more abstracted forms, and control flows downward [Bond 88]. This organization is very sensitive to failure in one of the high-level agents. The different agents do only have to know about their "mother" agent, and not about the other agents. Each agent gets his tasks from the "mother", processes the tasks, and gives then the result(s) back to the originator.

- **Heterarchical organization;** or organization as a community with agents having *rules of behavior*. It may be viewed as a community of interacting agents. Heterarchical structures, or flat structures, are much more robust to failure in high-level agents than in a hierarchical structure. On the other hand, this interaction structure often increases the communication and control overhead. There are not different levels of responsibility. All agents have to know about the other agents, and we have an all-to-all communication pattern.
- **Authority structure;** this type of organization reduces coordination work because agents with lesser authority must accept a goal or a result from an agent with higher authority. Thus, the most accurate and reliable agents should be allowed to guide the other agents. The need for authority organizations can be reduced by lowering the coupling and interdependencies among agents because conflicts will be reduced, and hence, the need to resolve problems with authority [Bond 88].
- **Marketlike organization;** Several researchers have proposed that distributed systems can be organized as markets. The Contract Net architecture, for example, supports a market in both task and processing resources, through competitive bidding [Bond 88].

However, several theorists of human organizations pointed out long ago that no organizational structure is appropriate in all situations [Bond 88]. An organization must be chosen for the task at hand. When choosing the type of organizational model, two aspects concerning task description should be focused; complexity and uncertainty. There are three types of complexity; information, task and coordination. Information becomes too complex when it needs more processing power than available to be processed. The information complexity can be reduced by abstraction and omission [1] [Fox 81]. Task complexity deals with the number of actions necessary to execute the task properly. The last one is coordination complexity. The key here is that the actions of each agent must be coordinated to produce the expected result as fast as possible. Uncertainty has also been recognized as a primary factor in choosing the right interaction structure among the agents [Fox 81].

Since all organizations have their drawbacks, it is often a good idea to decompose a particular problem into a combination of substructures. Vital parts of the network may have a heterarchical structure to increase the possibility of a solution from that part of the network.

Aspects of interagent communication

In order to coordinate activities, entities must communicate with each other. Communication provides the basis for all interaction. A bad communication strategy will therefore affect the performance of the interaction process. It is of vital importance that the communication strategy carry as few "bad habits" as possible to the overlaying strategies. Communication can be used poorly or efficiently, and used poorly it might lead to worse performance than with individual problem solving.

[1] Only communicate necessary information

Communication is used in order to provide agents with necessary knowledge to relate their actions to other agents' actions, to help synchronize actions and to provide agents with information generated by other agents [Bond 88]. An agent can use two methods to get the wanted information:

- by asking other agents
- by computation. This may not always be possible due to lack of necessary basic information at the agent or lack of processing power.

Two basic motivations for communication has been proposed [Bond 88]; agents share relevant information about the world they are sensing, and agents share relevant information about their own internal states. To decide what information is relevant is not straight forward.

Interaction between agents in a distributed environment is very expensive because communication is much slower than computation. This is certainly true if the agents are loosely-coupled. After deciding what to communicate, the information need to be packed into a suitable format. The receiving agent has to unpack the received packet, interpret the information and possibly respond. It is therefore very important to develop communication protocols that both minimize the need for communication among agents, and the amount of information that has to be communicated. When designing interactions among agents, at least the following issues should be taken into consideration [Hern 88]:

- not saturate communication channels
- use of efficient protocols. This will reduce the time needed for communication. Using large messages will reduce the communication overhead, but the messages will be more difficult to handle.
- problem decomposition. If sub-tasks are dependent a large communication overhead will be required to maintain global coherence. Thus, fine-grained decomposition will increase communication overhead.

It is important to note that limited communication resources should be used "intelligent" to improve problem solving. It is bad policy both to communicate too much or too little. In particular, too much communication should be avoided because it might lead to the following effects:

- delay on the communication channel due to heavy traffic
- PS efficiency decreases because agents sit idle waiting to communicate
- agents PS efficiency decreases because of the computing effort needed to transmit and receive messages
- it might cause *set effects*. Redundant information may confuse some agents.

The latter case needs some special attention. Let us consider the DSS[2] for tracking vehicles through a monitored area. An agent's detection is based upon signals received through acoustic sensors. These sensors are very sensitive to noise, so the probability of detecting a "ghost" vehicle is high. Thus, the agents have to cope with a large degree of uncertainty. The agents may cooperate and help each other by telling the neighbour agents when a tracked vehicle is moving from one region to another. If the agents transmits tracked vehicles uncritical, i.e. without being positive that it is a real vehicle, a heavy burden will be put on the system. First, lots of traffic will be put on the communication channels due to the agents transmitting unreal observations. Lot's of power will get lost because of agents transmitting and receiving information. Second, the agents attention will be transferred

[2] DSS: Distributed Sensor System

from monitoring the whole area to checking the correctness of receiving tracks. If the probability of transmitting a real track is high compared to transmitting a "ghost" track, this *set-effect* is positive. It will be a great help for the agents. On the other hand, if this probability is low, the result might be a "confused" network.

In a DPS network it is difficult to limit the interagent communication and still achieve the potential benefit from the distributed architecture. However, what and when to communicate is very important for the network performance. This leaves a heavy burden on the network coordination policies.

Uncertainty in a DPS environment

The need for limited communication in a DPS network makes coordination strategies very difficult to design [Corkill 83]. The result of limited communication is that each agent has a restricted view of the activity in the network. Thus, the result is different kinds of uncertainty within the network [Lesser 87], [Bond 88]:

- **environmental uncertainty;** agents does not have an accurate view of the other agents (number and skill), the sensors (location and characteristics) and the communication channels (location). This may result in poor estimation of changing environment.
- **data uncertainty;** an agent do not have consistent and accurate data available or they do not know the correctness of the data.
- **control uncertainty;** an agent does not have a consistent and accurate view of the other agents activity and they may also lack knowledge about the outcome of decisions.
- **behavioral uncertainty;** agents may not keep their delivery commitments.

Dealing with all these kinds of uncertainty is a major challenge to DPS systems. Once these problems have been solved, theory and solutions will be available which increase the reliability in a given network. A network handling the problems mentioned above, will also be able to handle lots of other problems (hardware failure, etc) given the right configuration. Coordination strategies must be developed for DPS systems in order to deal with these problems.

Local agent sophistication

It is unrealistic to believe that we can design network control policies which are sufficient flexible, efficient and that require limited communication resources, while making all the control decisions for all the agents in the network [Corkill 83]. It is therefore necessary to develop techniques which have a sophisticated form of local control.

When an agent is working alone it does not need to explicitly represent its current and expected future states. However, when an agent is to work together with other agents, it has to be able to represent these states explicitly. Both for its own use and for the use of the other agents. It is interesting to note that agents need more *self-awareness* when cooperating with other agents than when they work alone [Durfee 88]. An agent working alone can explore possible solution paths in an erratic manner, but when working with others it has to be more *predictable* for others to anticipate its actions and to coordinate interactions [Durfee 88].

In a distributed environment with a large degree of uncertainty the difficulty of controlling becomes pronounced [Durfee 88]. Control must be distributed. If not, it might result in:

- the reliability of the system decreases. If the *controller* agent fails, so does the system.
- the *controller* agent becomes a bottleneck because the request for decisions arrives faster than the *controller* is able to process them.

One of the things we require from a DPS network is reliability and graceful degradation. This requirement precludes the use of a global "controller" agent. In addition, a centralized control strategy is less flexible and less capable of taking maximum advantage of distributed processing resources [Hern 88].

Since control must be distributed, cooperation will require two types of control decisions [DurfeeB 87]:

- network control
- local control

Network control deals with problem decomposition and sub-task assignment, and has been the focus of much research within DPS. Most of the work has been concentrated on task-exchange algorithms (for example bidding) [DurfeeB 87].

Local control, on the other hand, deals with the agents local decisions. What sub-task to execute next, etc. Each agent must have sufficient knowledge to make its own decisions about sub-tasks to solve and solutions to communicate, and be able to evaluate its potential actions by taking the effect on network problem solving into consideration. How a given problem is decomposed into sub-tasks will affect the local control procedures. Examples of this includes the following [DurfeeB 87]:

- task precedence constraints; The local control decision of an agent with succeeding tasks will depend on the decisions made by the agent with the preceding task.
- replication; If equivalent tasks are assigned to several agents, local control decisions will depend on whether a equivalent task is executed or not.
- several pieces of a program is present. The need to synchronize these pieces, and to exchange information between them, will affect local control decisions.

These types of interactions are present in a DPS network. Thus, sophisticated local control at each agent is paramount for coherent cooperation. So far, the majority of local control techniques have been unsophisticated [DurfeeB 87].

As stated earlier, the agents local viewpoints do not have to be compatible with the network point of view. This, together with the need for local control at the agents, gives birth to a new problem. If the agents local control gets too "strong", it may result in *opportunistic* [3] behavior in the network. One reason for such a behavior may be that one agent has important knowledge that the others doesn't have. A cooperation strategy must make sure that things like this do not happen, and it will require a balance between local and global control.

Uncertainty and the distribution of control gives birth to another main concern; how to achieve network-wide coherence.

Coherence in a DPS environment

Coherence within a distributed environment can broadly be defined as [Durfee 85](pp. 1025):

[3] Opportunism: Agents take advantage of opportunities and circumstances, with little regard to consequences.

"the activities of the agents should appear to make sense given overall network goals."

Along some dimension of evaluation, *coherence* is a measurement on how well the system behaves as a unit. The following dimensions can be examined to evaluate coherence on a DPS system [Bond 88]:

- **The quality of the solution;** how satisfactory are the solutions produced, and how is the quality of them.
- **Efficiency;** how efficiently do the system produce solution(s).
- **Clarity;** how easy is it to describe and represent the system's actions to an outside observer.
- **Graceful degradation;** how does the system behave under failure, uncertainty and at the limits of its environment.

When efficiency is used as a measurement, incoherence can be the result of conflict over resources, agents duplicating each other's work redundantly or agents working against each other by undoing the results of another. This may be done unwittingly or maliciously [Bond 88]. According to [Fox 81], complexity may also reduce coherence. When tasks demands more resources than the current capability bounds, the need to allocate these adds additional work and uncertainty, which may lead to worse coordination. Coordination and coherence are related so that better coordination may increase coherence through the reduction of redundant work [Bond 88].

Global coherence is particularly difficult to achieve in a distributed environment where [DurfeeB 87]:

- communication is limited due to bandwidth limitations.
- each agent has only a limited view of the network activity (this is partly because of limited communication).
- there is no controller agent for coordination.

Obtaining global coherence is a key problem in a distributed problem solving network [Corkill 83]. If this global coherence is not achieved, the network performance can be diminished because of several factors. These factors include [Corkill 83]:

- processing power is wasted because agents wait for something to do
- processing power is wasted because agents work against each other i.e they work at cross-purposes with one another
- processing power is wasted because work is unnecessary duplicated
- processing power is misallocated

Factors affecting coherence

Achieving network-wide coherence is a quite complicated process. There are lots of parameters which have to be taken into consideration. How the task is decomposed will influence coherence in a DPS system. If it is decomposed so that dependencies among subtasks is reduced as much as possible, then there is less possibility for harmful interaction among agents. Interagent dependencies can also be reduced by adding more resources [Fox 81]. Regulation of resources is also a method of increasing coherence.

The content of communication is important both for cooperation and coherence within a DPS system. There are three characteristics of communication content in a distributed environment which have impact on coherence [DurfeeA 87](pp. 39):

- *relevance*; the degree to which the information is consistent with the solution derived

by the network. High relevance implies more coherence because work is stimulated along the solution path.

- *timeliness*; a measurement for the impact a message has on the receivers current activity. This is dependent on both the content of the message and the state of the receiving agent. Receiving a message with the answer to a task an agent is about to begin has high timeliness. If the message will have no effect on the receiver, there is no reason to transmit it. However, if the message will have a positive effect on the receiver (push the agent to work in more promising areas or the information is highly needed), it should be sent at once. Thus, it affects coherence.
- *completeness*; measures the fraction of a complete solution that the message represents. The bigger fraction, the higher completeness of the message. High completeness increases the coherence by reducing redundant messages, and redundant activities in agents. In addition, high completeness of messages reduces network traffic.

These three characteristics are not independent. For example, increasing completeness and relevance may decrease timeliness. Thus, compromises must be made when designing communication policies to increase coherence. An agent's focus can also be shifted because of interaction with other agents. This *distraction* can be both positive and negative. When it is positive the agent's focus is shifted to be more globally useful, while negative distraction introduces redundant activities [Bond 88]. As stated earlier, coordination has great impact on coherence. Effective coordination implies some degree of mutual predictability and lack of conflict. The more conflicts which have to be solved, the less well coordinate the agents. For a network coordination policy to be successful some criteria has to be met [Corkill 83](pp. 749):

- *coverage*; all portions of the overall problem must be included in at least one agent's activity
- *connectivity*; agents must interact in a manner which permits activities to be developed and integrated into overall solutions
- *capability*; the above mentioned criterias must be achieved with the communication and processing resources of the network

To ensure coherence each agent has to have an accurate and complete view of past, present and future activities of all the other agents. This implies simultaneous broadcasting of all state changes for all agents. Communication channels have limited bandwidth, they are not error-free and they introduce delays. Thus, there is no practical way to ensure coherence at a given moment [DurfeeB 87]. Instead, coherence will depend on how each agent make coherent local decisions based on its local view [DurfeeB 87].

Coherent behavior among agents in a DPS environment is crucial to achieve the potential network performance. However, much more research is needed in this area to understand how to achieve optimal coherence.

Concluding remarks

Distribution is an intrinsic characteristic of many of the computational and symbolic processing phenomena in the real world. It seems therefore natural that we pay attention to some of these phenomena when trying to design DPS systems. Both humans and computers have common characteristics; different *appropriateness* for a given problem, *limited capacity* (bounded rationality), *different skills*, etc. This paper has focused on two real world phenomena, scientific communities and organizational theory, and tried to show

that some aspects of them may be applicable as metaphors when designing DPS systems. Scientific communities; when agents needs to be "intelligent", independent and their relationship is changing relatively often. Organizational theory; when the individual agents have well defined roles and their relationship is relatively static. However, in the real world, a combination of the two may be the solution to some problems.

At the time being, metaphors from organizational theory seem most appropriate. The individual agents need less "intelligence" because of their relatively static relationship and they are therefore much easier to model. However, a future goal must be to develop techniques sophisticated enough to handle cases were scientific communities are applicable as metaphors.

Emphasis in this paper has been put on DPS domains where the relationship among the agents is well defined, and thus motivated by the organizational theory metaphor. Cooperation among the agents has been identified as a key issue within DPS research, and some of the main concerns when designing such a system have been addressed. Cooperation is what gives DPS systems their potential power. However, cooperative behavior among the agents are not easy to achieve. When deciding the interaction structure among the agents, several aspects must be taken into consideration. Several structures have been discussed, and it is often a good idea to use a combination of different structures. As concluded, communication is fundamental in a DPS system. Unfortunately, the communication channels are often limited, so agents have to be selective about what and when to communicate. This restricts each agent's view of network activity and the result is different kinds of uncertainty. Techniques that are able to handle these kinds of uncertainties, also have the potential of making general networks more robust to failure. The distributed environment, with the mentioned characteristics, stresses the control problem. Since control also has to be distributed, the agents have to be more sophisticated. Agent sophistication turns out to be an important research issue, and especially in domains were the scientific community metaphors seems suitable. One of the major challenges is to make agents smart enough to take decisions that are coherent within the network, despite their lack of knowledge about the "world". I don't think that it is possible to achieve global coherence within the network at a given time. It is far too many factors that have influence on coherence. Our goal should therefore be to achieve *sufficient* coherence in the network. This means to design DPS systems that acts as a team and perform significantly better than agents working on their own.

This paper is part of my Cand.Scient thesis at the University of Tromsø. The thesis deals with DPS in general, and with emphasis on coordination. Aspects discussed in this paper is used as background material for cooperation and applied to a case-study in my thesis; - a system for monitoring patients during surgery.

Acknowledgement

The author wishes to thank Sigmund Akselsen og Thore Danielsen for valuable comments on earlier versions of this paper.

Appendix A.
References

[Bond 88] Alan H.Bond and Les Gasser
Readings in Distributed Artificial Intelligence.
Morgan Kaufman, 1988.

[Corkill 83] Daniel D.Corkill and Victor R.Lesser
The use of meta-level control for coordination in a distributed problem solving network.
In Proceedings of the Eighth International Joint Conference on Artificial Intelligence:748 - 756, August, 1983.

[Durfee 85] Edmund H.Durfee, Victor R.Lesser and Daniel D.Corkill
Increasing coherence in a distributed problem solving network.
In Proceedings of the Ninth International Joint Conference On Artificial Intelligence:1025 - 1030, August, 1985.

[Durfee 88] Edmund H.Durfee
Coordination of Distributed Problem Solvers.
Kluwer Academic Publishers, 1988.

[DurfeeA 87] Edmund H.Durfee, Victor R.Lesser and Daniel D.Corkill
Cooperation through communication in a distributed problem solving network.
In Michael N.Huhns (editor); Distributed Artificial Intelligence:29 - 58, 1987.

[DurfeeB 87] Edmund H.Durfee, Victor R.Lesser and Daniel D.Corkill
Coherent Cooperation Among Communicating Problem Solvers.
IEEE Transactions on Computers C-36 No. 11:1275 - 1291, November, 1987.

[Fox 81] Mark S.Fox
An organizational view of distributed systems.
IEEE Transactions on Systems, Man and Cybernetics SMC-11 No.1:70 - 80, January, 1981.

[Hern 88] Luis E.C.Hern
On distributed artificial intelligence.
The knowledge engineering review 3(1):21 - 57, March, 1988.

[Kornfeld 81] William A.Kornfeld and Carl E.Hewitt
The Scientific Community Metaphor.
IEEE Transactions on Systems, Man and Cybernetics SMC-11 No.1:24 - 33, January, 1981.

[Lesser 87] Victor R.Lesser and Daniel D.Corkill
Distributed Problem Solving.
In *In Encyclopedia of Artificial Intelligence*, pages 245 - 251. John Wiley & Sons Inc, 1987.

[Simon 57] H.A.Simon
Models of man.
New York: Wiley, 1957.

[Smith 80] Reid G.Smith
The contract-net protocol: High-level communication and control in a distributed problem solver.
IEEE Transactions on Computers C-29(12):1104 - 1113, December, 1980.