1

Learning Communication Strategies in Multiagent Systems

Michael Kinney[1] and Costas Tsatsoulis

Department of Electrical Engineering and Computer Science
The University of Kansas
Lawrence, KS 66045

tsatsoul@kuhub.cc.ukans.edu
(785) 864-4615
FAX: (785) 864-7789

1. Introduction

The issues that need to be addressed by research in multiagent systems include communication, coordination, coherence, uncertainty, organizational structure, planning, and knowledge representation. Our work addresses communication, coordination, and coherence, and, as we show in the following sections, our methodology allows the multiagent system to dynamically organize its structure and problem solving, thus making the issues of organizational structure and planning redundant.

More specifically, as with any other computing system, the fundamental goal of research in multiagent systems is to make the system have "good" performance. To achieve good performance an agent should not be idle, waiting to receive the information it needs to perform its task, should not duplicate the work of other agents (with the exception of systems of agents competing for resources, as, for example, ETHER [Kornfeld 79], or systems where a high degree of redundancy is required), and should be assigned tasks appropriate to its abilities. At the same time each agent should perform activities that are globally coherent, that is, improve the problem solving state of the whole collection of agents [Bond and Gasser 88].

Most research in Distributed Artificial Intelligence (DAI) and multiagent systems has tried to solve these problems by addressing the communication needs of an agent: *what* information to send, *when,* and to *whom*. Even research that superficially solves a different problem, is primarily addressing the issue of communication between agents. For example, research in the topology of agents is motivated by the need to perform problem decomposition, and to determine communication links and activities. Research in agent coordination is primarily interested in communication of information and agents' models of other agents. Even research in an agent's knowledge representation is driven by the types of information communicated and the quality of the communication acts (see, for example [Fagin and Vardi 86]).

---

[1] Currently with DeskStation Technologies, Lenexa, KS.

In general, if an agent had complete knowledge of all other agents, their exact state at any time (that is, their information, knowledge, intent, and internal models of themselves and the other agents), and the overall goal of the system, then it could be perfectly coordinated with the activities of every other agent. Of course, this is infeasible, since the communication requirements would overwhelm the problem solver.

The problem of communicating information to achieve coherence has been approached in many ways: through studies of the effects of different agent organizations on the performance of multiagent systems; or by determining an applications specific communications strategy; or by including planning communications actions as part of the regular problem solving activities of each agent; or by providing a common area for groups of agents to store their knowledge (blackboard systems). Most approaches to communication between agents result in static organizations and strategies that are optimized for the specific application for which they are adopted. In our work we are interested in developing a methodology that allows the dynamic, adaptive generation of communication strategies for any collection of cooperating agents. To address the communication and coordination issue for cooperative, multiagent systems, we introduce an adaptive communications strategy. Instead of each agent following a static communications protocol, agents learn local communications strategies that improve the efficiency of each agent. These local communications strategies interact with one another resulting in a global communications strategy that improves the coordination and efficiency of the entire multiagent system.

Our basic approach is to provide an agent with the ability to adapt to the environment in which it is placed and the problem solving context in which it is involved. If the agents cooperate in this learning process, the performance of the entire multiagent system improves. This allows agents to adapt to different organizational structures, adapt to the specific needs of different applications, avoid being constrained by a predetermined plan of action, and eliminate the overhead of a common area for storing knowledge. It also leads to the dynamic establishment of communication strategies, agent organizations, and problem solving contexts.

In the rest of the paper we describe our methodological approach and the dynamic, adaptive communications strategy we developed. We discuss the behavioral parameters of each agent that need to be determined, and provide a quantitative solution to the problem of controlling these parameters. We also describe the testbed we built and the experiments we performed to evaluate the effectiveness of our theory. Several experiments using varying populations and varying organizations of agents were performed and are reported. A number of performance measurements were collected as each experiment was performed so the effectiveness of the adaptive communications strategy could be measured quantitatively.

The adaptive communications strategy proved effective for fully connected networks of agents. The performance of these experiments improved for larger populations of agents and even approached optimal performance levels. Experiments with non-fully connected networks showed that the adaptive communications strategy is extremely effective, but does not approach optimality. Other experiments investigated the ability of the adaptive communications strategy to compensate for "distracting" agents, for systems where agents are required to assume the role of information routers, and for systems that must decide between routing paths based on cost

information. Even though the adaptive communication strategy improved the efficiency of these systems, the results fell short of the optimal performance levels. The reasons for the degraded performance of these special cases are discussed, and possible improvements to the adaptive communication strategy are proposed.

2. Approaches to the Problem of Communication in Multiagent Systems

There is a large volume of previous research addressing coherence, communication and coordination in Distributed AI (DAI) systems, and in multiagent problem solving systems, going back to the early blackboard systems. The major issue of study has always been coordination of the problem solving activity, and different projects have attempted to answer the question from different perspectives.

One solution to the coordination problem is for agents to plan the actions that the system performs. Centralized and distributed planning allow agents in a distributed system to align and synchronize their activities towards a set of common goals. Among the issues concerning centralized planning are problem decomposition and plan generation, which, in turn, involve plan representation, plan execution, and plan evaluation [Rosenschein 82]. Classical frameworks like the Contract Net Protocol generate a cooperating plan by contractor-contractee task-assignment [Smith 80; Davis and Smith 83]. Distributed planning is an incremental approach which involves plan reconciliation, plan synchronization [Rosenschein 82], partial plan integration, and conflicting plans resolution [Georgeff 83; Lesser and Erman 80; Lesser and Corkill 83; Durfee, Lesser and Corkill 87; Durfee and Lesser 88]. Partial Global Planning [Durfee and Lesser 87; Durfee and Lesser 89; Durfee and Lesser 91] and its extension, Generalized Partial Global Planning [Decker 93], is a flexible coordination approach which does not assume any particular distribution of subproblems, expertise, or other resources; instead it lets the nodes coordinate by exchanging information about their objectives and plans. Through information sharing, nodes exchange their partial plans to gain a more global view. In a distributed planning system, local decisions always have nonlocal impacts [Conry *et al.* 89], so, in order to converge a set of local plans to a global one, both multi-stage negotiation and information sharing are necessary to the system planning. Another solution is offered by blackboard systems: the agents (knowledge sources) share all knowledge by posting it on the blackboard, thus always having a complete global view of the problem solving activity [Erman *et al.* 80; Hayes-Roth 85; Ensor and Gabbe 85]. Finally, one possible solution is not to communicate at all. Shoham and Tennenholtz's "social laws" can be viewed as an approach which tries to change the structure of the agent's tasks by, for example, restricting an agent's possible activities [Shoham and Tennenholtz 92]. In other research, agents do not communicate but, given a problem and a set of possible solutions from which the agents need to choose one, focal points are prominent solutions of the problem to which agents are drawn [Fenster, Kraus, and Rosenschein 95].

In our work agents transmit very low-level parameters, and there is no need for them to reason about their own subproblems, resources and expertise or these of their neighbors, i.e., agents connected to them directly. This simplifies the agent's knowledge and allows multiagent systems to use agents that are "savants", i.e. capable only of individual problem solving without

the need of social knowledge about the overall group, as, for example, proposed in [Shoham and Tennenholtz 95]. We also minimize the communications overhead by having the agents learn exactly what to send, to whom, and how often.

Other work has concentrated on the topologies required by DAI systems, since the structure of an organization provides a framework of constraints and expectations about the behavior of the agents in the distributed system. In comparisons between hierarchical and flat topologies it was found that both have their advantages and disadvantages. The best compromise is a topology that is mostly flat, so low level information can be shared, yet also has a hierarchy, so higher levels of information can be collected [Fox 81; Wesson *et al*. 81]. Other work approached the topology problem through mathematical models of the costs associated with different market topologies [Malone 87]. The agent topologies required by a distributed air traffic control application were studied in [Steeb *et al*. 81]. In [Cammarata *et al*. 83], it was pointed out that organization policies not only define a task decomposition, but also prescribe communication paths among agents.

Our work is not concerned with special agent organizations, since the system will adaptively organize itself. We assume a swarm as an initial agent organization to study the communications problem at its most basic, but our methodology is applicable to special agent organizations.

In a multiagent system, the ability of an agent to learn can be viewed as learning as a group, and learning to adjust its views or actions [Shaw and Whinston 90]. Most work has concentrated on the contents of the agents only, either the problem solving components or the shared view of the world or the vocabulary used. One learning system incorporated contract net bidding and genetic algorithms into a DAI framework. The contract net bidding process introduced competition between agents, and the genetic algorithms allowed agents to evolve into more efficient problem solvers. Adaptation occurred at a very high level, and took place over a long period of time as weaker problem solving agents were replaced by more efficient problem solving agents, taking advantage of an evolutionary model of learning [Shaw and Whinston 90]. In [Brazdil and Muggleton 91; Brazdil *et al*. 91] agents in a multiagent system start with different views of the world, and use different vocabularies to describe the world. Theory integration uses inductive learning to resolve their world views and vocabularies. This results in individual agents modifying their rule sets so the group of agents has a consistent view of the world. An architecture for agents in a multiagent system that combines machine learning and hypothesis sharing was proposed in [Smith and Davis 81]. Agents use an inductive learning process to build a generalization hierarchy. The generalized hypothesis is shared with other agents which incorporate this hypothesis into their own generalization hierarchies resulting in a distributed inductive learning system. In [Gallant 88] a DAI system is described where the components of rules are separated across nodes, and neural network techniques are used to adjust the weights of the connections between the nodes. This is a fine grain distribution of a problem, and the nodes are so simple that they can not be considered intelligent agents. This type of system also requires a large number of tightly coupled nodes with connections capable of high communication bandwidths. More recently, a methodology was developed to allow an agent to adapt its reasoning method, meta-control strategy, perceptual strategy, control mode, etc. to fit in a changing environment [Hayes-Roth 95].

All this work in learning agents, though, has concentrated on an individual autonomous agent. Our work applies learning to the behavior of a multiagent system, concentrating on communication rather than individual problem solving and reasoning components.

More recent work has started looking at communication schemas between agents. As pointed out in [Decker and Lesser 94], meta-level communication between agents about their local loads can, with a small communication cost, pinpoint the true costs and benefits of the various organization structures, allowing an informed organizational decision to be made. For example, agents in Dis-DSS (Distributed Dynamic Scheduling System) share their ability and availability with each other [Neiman *et al.* 94]. Dis-DSS is a protocol for the exchange and updating of resource profiles including agent's committed resources, available resources, and estimated future demand. In the DAI system described in [Turner 94], an agent selects information by making use of the agent's own problem solving knowledge, and the usefulness of information. Knowledge about how the information item was found and the role that it played in problem solving is translated into an expected value through a set of heuristics. Another technique used a scheduler which studied the current tasks and goals of the system and enforced a combination of up to five pre-defined coordination algorithms for multiagent systems [Decker and Lesser 95].

Most relevant to our work are recent projects where agents learn organization or communication strategies to function in a multiagent group. In the work by Prasad, Lesser and Lander [1996], different agents play different roles in the search towards a goal in a multiagent system and these agents learn, through supervised learning techniques, what role to play in the organization by estimating a search control strategy based on Utility, Probability, and Cost [Whitehair and Lesser 93]. The system LEMMING learns to reduce the communication load for task negotiation in a Contract Net by case-based reasoning. An agent uses CBR to decide to which other agent it should sent the Task Announcement messages which are part of the Contract Net communications protocol. The cases are selected based on the current task the agent is trying to complete [Ohkno, Hiraki and Anzai 96]. In a brief paper, Foisel, Chevrier and Haton [1996] mention that agents initiate interactions with other agents and, after the interaction is finished, they evaluate the benefits and costs of these collaborations using *a priori* defined evaluation criteria. Interactions with agents which led to positive results are reinforced, while interactions with agents which led to negative results are weakened. In other work, interaction between agents was represented as a two-player game where the agent's objective was to look for a strategy that maximizes the expected sum of rewards in the game. The goal is to learn the other agent's model, so its future behavior can be predicted [Carmel and Markovitch 96]. Similar in goal was the work described in [Bui, Kieronska and Venkatesh 96]: learning allowed agents to learn about other agents' preferences via past interactions in a communication-intensive negotiating agent architecture. In this environment agents negotiate iteratively and *refine* their preference functions until a consensus is reached. Agents try to guess the preference functions of other agents, and the guess is updated as more observed behavior of the other agents becomes available.

Most of this work aims to develop learning techniques for agent models or types of interaction, and, while the results of the adaptive re-organization of the multiagent system will influence the communications strategy indirectly, the explicit goal of the learning agents is not an improved communications strategy. The work most relevant to ours is the one implemented in

LEMMING, where agents learn who to talk to and what information to transmit. LEMMING uses case-based reasoning so that an agent will remember which agents were useful in previous contract negotiations and will prefer communicating with them; in other words the agents learn based on whether the information received was useful to them or not. Our work has similar goals, i.e. reduction of the overall communications load, but we take a more global view: agents consider their own information needs, but they also act as routers of information, thus considering the information needs of the other agents in the network, too. Also, LEMMING is constrained to a Contract Net architecture, while our system develops a generic, domain- and organization-independent adaptive communications strategy.

3. An Adaptive Communications Strategy for Multiagent Systems

3.1. System Components

For the purposes of our work a multiagent system consists of sensors and agents connected by links through which information can be transmitted. A sensor is a source of information, and the agent uses information collected from other agents and sensors to perform problem solving activities.

All agents are assumed to be cooperative, and working on the same problem. Every agent is assumed to have the same communications capabilities, and use a common information syntax. This assumption is required for agents to share information. The only component of an agent that is allowed to vary is the agent's specific problems skills. This means that agents can be experts in specific areas, but they are required to cooperate with one another to solve a given problem.

Figure 1 shows the layered architecture for an agent: the agent has a problem solving component, a low-level communications component that allows it to talk to other agents and sensors, and an adaptive component, which allows it to learn communication strategies (this is discussed in detail starting in section 4). In this specific example the problem solving component of the agent is rule based. A_i represents a connection to a neighboring agent, and S_j represents a connection to a neighboring sensor.

```
┌─────────────────────────────────────────────────┐
│  ┌─────────────────────────────────────────┐    │
│  │       Problem    Solving                 │    │
│  │            Component                      │    │
│  │  ┌──────────────┐                         │    │
│  │  │  Rule Base   │                         │    │
│  │  └──────────────┘                         │    │
│  │         │                                 │    │
│  │         ▼   Rules                         │    │
│  │  ┌──────────────┐      ┌──────────────┐   │    │
│  │  │  Inference   │◄─────│  Fact Base   │   │    │
│  │  │   Engine     │      └──────────────┘   │    │
│  │  └──────────────┘                         │    │
│  └─────────────────────────────────────────┘    │
│         │                        ▲               │
│         ▼ Conclusions            │ Facts         │
│  ┌─────────────────────────────────────────┐    │
│  │  Adaptive Communication Control System  │    │
│  │  ┌────────┐  ┌────────┐  ┌────────┐      │    │
│  │  │Model of│  │Model of│  │Model of│      │    │
│  │  │ Sensor │  │ Agents │  │ Itself │      │    │
│  │  └────────┘  └────────┘  └────────┘      │    │
│  └─────────────────────────────────────────┘    │
│         │                        ▲               │
│         ▼ Send                   │ Receive       │
│           Messages                 Messages      │
│  ┌─────────────────────────────────────────┐    │
│  │   Low    Level   Communications          │    │
│  │            Component                      │    │
│  └─────────────────────────────────────────┘    │
└─────────────────────────────────────────────────┘
```
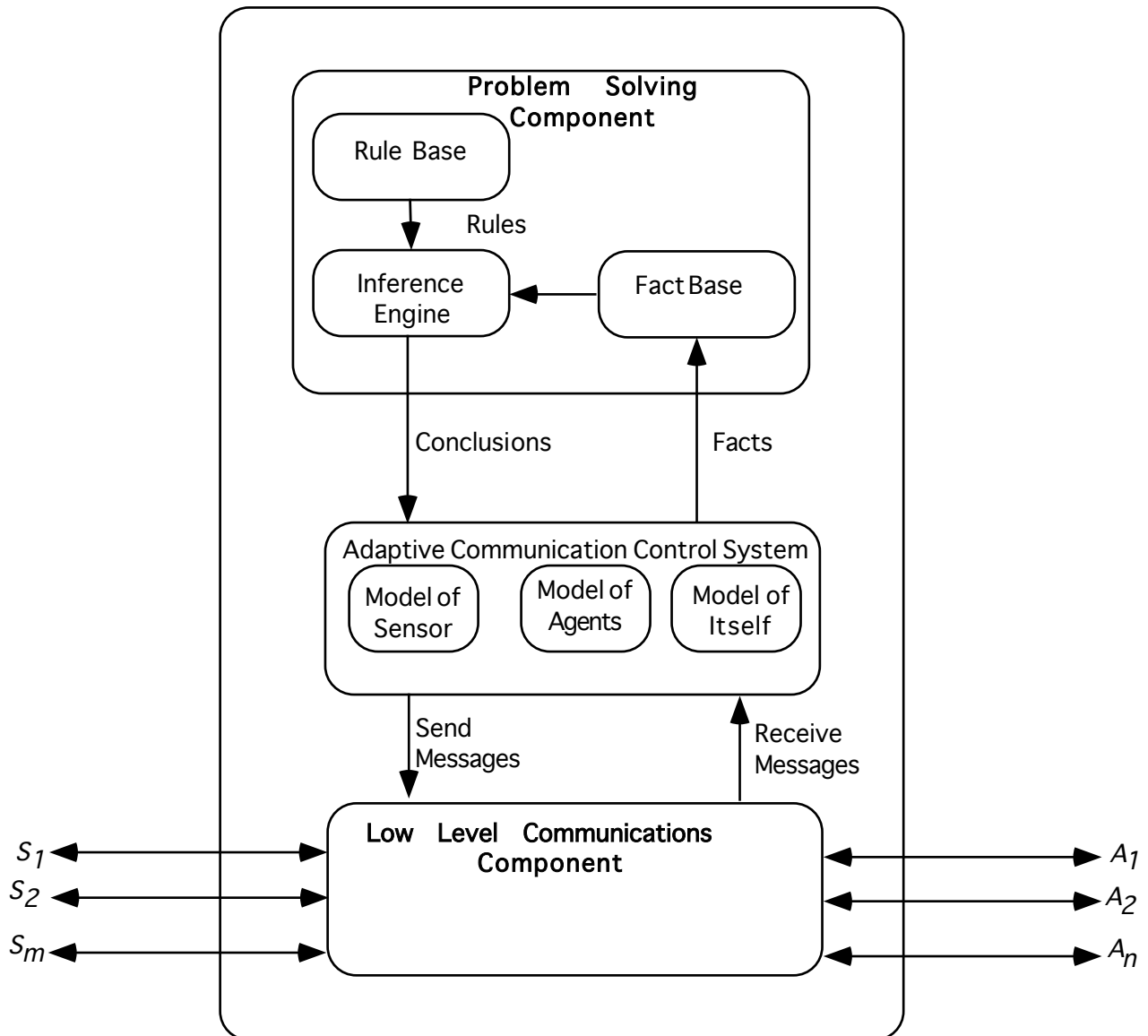
Figure 1. Layered Agent Architecture with a Rule Based System

We assume a flat agent organization; this makes communication and coordination more difficult, since there is no hierarchically "superior" agent to guide and control activities of lower-level agents. We selected a flat organization to make sure we are addressing the communication and coordination problem at its more difficult level. Our work can be easily applied to hierarchical agent organizations since we allow agents to have specialized functions (e.g. problem solvers, organizers, information routers, problem decomposers, etc.) and our links can transmit information between agents at any levels of a hierarchy.

3.2. Agent Behavior

Each agent is required to make local decisions that help the cooperating group of agents move towards a solution to a given problem. Since each agent has information about itself and a small amount of information about its neighbors only, making good decisions is difficult. The behavior of a single agent is similar to that of a pleasure seeking automaton [Jacobs 1971]. Instead of looking for food and avoiding predators, the goals of a pleasure seeking automaton in a problem solving environment are to collect information that helps the group of agents solve the given problem, and, at the same time, to perform its own problem solving activities. This means that information that effects an agent's own problem solving activities and information that might effect the problem solving activities of its neighbors must be collected. By correctly balancing the collection of these two types of information, an agent can perform both problem solving and information routing. In addition, each agent is required to collect information from the least expensive sources possible. The actions of sending information and problem solving have costs associated with them. As information propagates through a system, it has a cost attributed to the actions that were performed for it to reach its current location. If there are multiple paths through which a piece of information can be sent or generated, the agents that use that information must favor the least expensive path to reduce the communication load and to improve the overall efficiency of the problem solving system. Note that there are application where the quality of the information may be more important than the cost. In this paper we are only concerned with information cost, while other work has looked into quality and reliability of information [Tsatsoulis and Yee 96].

For agents to learn the information needs of their neighbors and find the least expensive path for routing information, they must provide feedback to their neighbors concerning their own information needs. Once this feedback has been collected, an agent can make intelligent decisions concerning the routing of information. Another type of feedback that is required is the work load of neighboring agents. If a neighboring agent is underworked, then more information should be sent to that agent. If a neighboring agent is overworked, then less information should be sent to that agent. The combination of these two types of feedback provides the information required by agents to make local decisions that effect to global flow of information through a network.

The final aspect of an agent's behavior is how it budgets its time towards problem solving activities, communicating with neighboring agents, and collecting information from sensors. The amount of time spent on each of the three major activities is critical to the performance of an agent. If too much time is spent collecting information from sensors, the information will overflow the system, and very little problem solving will be performed. If too much communication is performed, information will be propagated throughout the network, but once again, not enough problem solving will be performed. If too much emphasis is placed on problem solving, an agent will perform all the problem solving possible. This will leave it in an idle state as it waits for more information from neighboring agents or sensors.

Since the balance among these three activities is not normally the same for every agent in a system, a mechanism must exist to adjust the importance of each activity. A simple solution is to rely on the feedback concerning the work load of neighboring agents to adjust the flow of information into an agent. If the flow of information into an agent is increased, that agent is forced to spend more time communicating and less time problem solving. If the flow of

information into an agent is decreased, it has more time to spend on problem solving. The control of time spent collecting information from sensors can also be adjusted by examining the work load of an agent. If an agent recognizes that it is overworked, it will not only slow down the flow of information from neighboring agents, it will also slow its own collection of information from sensors. If an agent is underworked, it will request an increase in the flow of information from neighboring agents, and spend more time collecting information from sensors.

The behavior of a single agent in a cooperating group can be summarized as follows:
1) Collect information that effects your own problem solving activities.
2) Collect information that might effect the problem solving activities of your neighbors.
3) Collect the least expensive information possible.
4) Share information that might effect the problem solving activities of your neighbors with your neighbors.
5) Share your information needs with your neighbors.
6) Share your current work load with your neighbors.
7) Budget time between problem solving, communicating with neighboring agents, and collecting information from sensors based on your own work load and the work loads of neighboring agents.

This behavior needs to be dynamic, and adapt itself to different problems and configurations of the agent network. It is possible, for example, that different problems will use different parts of the network, requiring that the agents adapt their behavior to achieve better overall network performance. It is also possible that some agents or links between agents might disappear, requiring that the rest of the network dynamically adapt its behavior to the changes in its topology. What is needed is an *adaptive communications strategy* to modify and adapt an agent's communications activities as it interacts with other agents in a multiagent system. If all the agents in a multiagent system use an adaptive communications strategy, the collection of local communications strategies (i.e., the communication activities of an agent with its immediate neighbors) represents the global communications strategy that controls the flow of information through a network of agents.

3.3. The Communications Layer of an Agent

An adaptive communications strategy optimizes an agent's local communications strategy so it can take advantage of its local resources. In addition, the adaptive communications strategy can compensate for failures in agents and communications links. An ideal communications strategy uses the minimum amount of communication required for a group of agents to solve a problem in the minimal amount of time. The goal is to design a separate communications layer that has the ability to adapt to the problem solving environment in which it is placed. This requires that the communications strategy be independent of the problem domain, independent of the network topology, able to adapt to the information needs of the agents, and able to minimize communications with other agents.

Since this communications layer is designed to be used in a variety of environments, it will not be as efficient as a communication strategy designed for a specific application. However, the

advantages of an adaptive communications strategy outweigh its limitations. Since an adaptive communication strategy learns, an application can be implemented in less time. Also, the number and types of agents in a multiagent system can be adjusted without redesigning the communications software. In addition, the learned communications strategy can provide a starting point for implementing an optimized communications strategy. The areas that an adaptive communications strategy shows the most promise are in prototyping MAS applications and in any dynamic problem solving environment where the communications strategy must change over time.

The communications layer is modeled as a set of behaviors required by an agent in a multiagent system. This set of behaviors is also modeled as a control system with a set of inputs, a set of outputs, and a set of system equations.

4. The Adaptive Communications Control Strategy (ACCS)

The mechanism used to achieve the seven behaviors listed in the previous section must be simple so the overhead associated with the adaptive communications strategy does not dominate the work that an agent performs. This mechanism is a set of probability tables that are used to make information routing and sensor sampling decisions. Each agent is a control system with a set of inputs and a set of outputs. The inputs include information from agents, information from sensors, and feedback from neighboring agents. The outputs include information to other agents, feedback to other agents, and sensor sampling actions. Inside the agent is a system of equations that adjusts the relationship between these inputs and outputs in an attempt to satisfy the agent's behavioral characteristics.

This system of equations also generates two probability tables. One is used to make information routing decisions, and the other is used to make sensor sampling decisions. As information is received from external sources or generated by the problem solving component of an agent, it is classified into independent sets. Knowledge is grouped in this way to provide an agent with a finer level of control over its information routing and sensor sampling decisions. The information routing probability table contains a row of values for each neighboring agent. Each row contains a value for the different classes of information. When information is received from an external source or information is generated by the problem solving component of an agent, the information is classified. The column of the probability table associated with that information class is then examined. Each probability value represents the frequency with which information of a specific class should be transmitted to an adjacent agent.

The sensor sampling probability table is used in a similar manner. Each time an agent decides to perform a sensor sampling action, this table is examined. The probability values in this table represent the frequency with which information of a specific class should be sampled from neighboring sensors. The control system within an agent adjusts the probability values in each of these tables to achieve the desired agent behaviors. The probability tables used by all the agents in the system are a direct representation of the current communications strategy. This strategy is adaptive because the values in these tables are adjusted over time in an attempt to improve the performance of the system (see sections 4.6, 4.7, and 4.8 for details).

Several topics must be introduced before the system of equations within an agent can be described. These include the classification of information, the usefulness of information, the usefulness of classes of information, the cost of information, the selfishness of an agent, and the work load of an agent. After these topics are discussed, they will be used to describe the system of equations that generate the probability tables that an agent uses to make information routing and sensor sampling decisions.

4.1. Classes of Information

Information must be grouped into separate classes so an agent can make better decisions. If no information classification is performed, feedback would be required for each piece of information. However, since each piece of information is unique, the agent that generated the information will not have a chance to use the feedback information to make a more intelligent routing decision the next time. This leaves an agent without any mechanism for learning, and it must resort to sending all new pieces of information across all possible links so the neighboring agents are guaranteed of acquiring the information they need to perform their part of the problem solving.

On the other hand, if a single classification is used, an agent is only able to tell a neighbor the usefulness of all the information that it has received. Since most agents have specific problem solving skills, they only need a small subset of the information. This results in relatively small feedback values, and thus small probability values, decreasing the frequency with which information is transmitted, and the agents are idle most of the time.

The first level of classification that is required is to separate information into groups based on who sent the information. This way, an agent can track the information that is passed across specific links. Then, through feedback, the traffic across specific links can be adjusted. This helps strengthen or weaken certain paths through the network of agents.

The next level of classification depends on the information needs of the problem solving components of the different agents. The assumption is made that a common set of classification rules exists for all agents and that all generated information will fall in one of these predefined classes. This assumption does not hurt the generality of our approach, since it is reasonable to assume that the user or designer of a problem solving network will know *a priori* the knowledge that the network can deal with.

4.2. Usefulness of Information

As an agent collects information, it groups it by class and by the agent that sent the information. These groups of information are evaluated to determine how useful they are to the problem solving component of an agent and to the neighboring agents. This means that the Adaptive Communications Control System must know what the problem solving component of an agent considers useful. Once the usefulness of an information class has been determined, the

probability tables can be computed and fed back to the neighboring agents (see sections 4.6, 4.7, and 4.8 for details). This feedback provides each agent with a model of what its neighbors consider useful and not useful. The same method is used to examine the information sent by neighboring sensors, so that an agent has a model of which sensors provide information that is useful to the system of agents.

The usefulness of information must be examined at two different levels. One is the usefulness of a single piece of information, and the other is the usefulness of a class of information. The usefulness values for the pieces of information in a class are combined with the usefulness of the information class to neighboring agents to determine the overall usefulness of a class of information. The usefulness of a single piece of information will depend on the specific needs of the problem solving component of an agent.

4.3. Cost of Information

There are two actions that have an associated cost in a system of agents. One is computation, and the other is communication. As a piece of information is passed across communication links, its cost increases by the cost of using these links. Also, any piece of information that is generated by the problem solving component of an agent has a cost attributed to the cost of performing the problem solving activity. As a piece of information propagates through a system of agents, the cost associated with that information is a direct representation of the resources that the information consumed.

The cost is used to help an agent make better information routing decisions. The less expensive routes should be preferred over more expensive routes. However, to increase the robustness of the system, routes should not be completely cut off. Even though one route is more expensive than another, if the less expensive route is lost due to an agent or a link failure, the more expensive route will be needed. If the more expensive route is completely cut off, the system would not be able to adapt when the failure occurred. This compromise of allowing some amount of traffic to pass through expensive routes may decrease the performance of the system, but it is required in order to increase its robustness and adaptability.

4.4. Agent Selfishness

One behavior that an agent possesses is the ability to balance its own information needs against the information needs of its neighbors. If an agent is completely selfish, it will collect information that it needs for its own, local problem solving, but it will not provide any information that its neighbors may need. If an agent is completely selfless, it will spend all of its time routing information to its neighbors, and spend no time problem solving. Though there are special cases where completely selfish and completely selfless agents are needed, most of the time a combination of these characteristics is required.

To address this problem, we introduce the concept of a *selfishness factor,* having a range [0,1]. A value of 0 means that an agent is completely selfless, and a value of 1 means that an agent is

completely selfish. For most cases, a selfishness value of between 0.5 and 1 will be used. This range of values places a higher importance on an agent's own information needs, but it forces an agent to also consider the information needs of its neighbors. The set of equations that determine the behavior of an agent must use this selfishness factor to balance the activities of problem solving and information routing.

4.5. Agent Work Load

An agent also monitors its own work load. This information is used to adjust the amount of information that neighboring agents send and the frequency by which neighboring sensors are sampled. If an agent is underworked, the neighboring agents should send more information, and the neighboring sensors should be sampled more often. If an agent is overworked, the neighboring agents should send less information, and the neighboring sensors should be sampled less often. This leads to several questions: what is the definition of "work"?; what is considered "underworked"?; and what is considered "overworked"?

The goal is for all the agents to be working at maximum efficiency, which means that they should fully utilize all their resources. However, it is difficult to tell the difference between an agent that is overworked, and an agent that is working at maximum efficiency; both are using 100% of their resources. To make this distinction, we use a definition of work based on an agent's *idle time,* and we demand that all agents functioning properly have some idle time. Consequently, if an agent is never idle, it is overworked; if an agent is always idle, then it is underworked; if an agent has only a small, predefined amount of idle time, then the agent is working at its maximum possible efficiency. With this definition of work, the difference between overworked and working at maximum efficiency can be determined.

Some agents, due to their location in the network, or their specific problem solving skills, will have more work to do than other agents. These agents are considered the "weak links" in the system, since the network of agents can only process information at the rate of its slowest member. These weak link agents will quickly become overworked, and the communications strategy developed should take into account such overworked agents, and make sure that they work at peak performance, without slowing down the network-wide problem solving.

It is clear that in some agent topologies it will be impossible to have all agents working at peak performance, especially given even a single weak-link agent. The best we can hope to achieve is that the weak link agent works with only a small amount of idle time, and the remaining agents have minimum idle time, which is computed based on the resources these agents need to contribute to the cooperating group. Therefore, for the multiagent system to operate efficiently, each agent has to have an expected amount of idle time that it should attempt to achieve. To achieve this goal, each agent contains a control system that adjusts the amount of resources an agent uses in an attempt to match the measured amount of idle time to the expected amount of idle time.

4.6. General Form of an Adaptive Communications Control System (ACCS)

The concepts of information classification, information usefulness, information cost, agent selfishness, and agent work load are combined in one set of equations that determine the behavior of an agent.   Figure 2 represents the control system model of an agent. The inputs are shown on the left side, the outputs on the right side, and the system of equations is inside the agent.

For this model, there are ***n*** neighboring agents and ***m*** neighboring sensors. Information enters an agent and is either used by the problem solving component to generate new pieces of information, routed to a neighboring agent, or discarded. New pieces of information generated by the problem solving component are either routed to a neighboring agent or discarded. K_A, K_T, ΔT and T_E are parameters to the internal control system that monitors the work load of an agent.
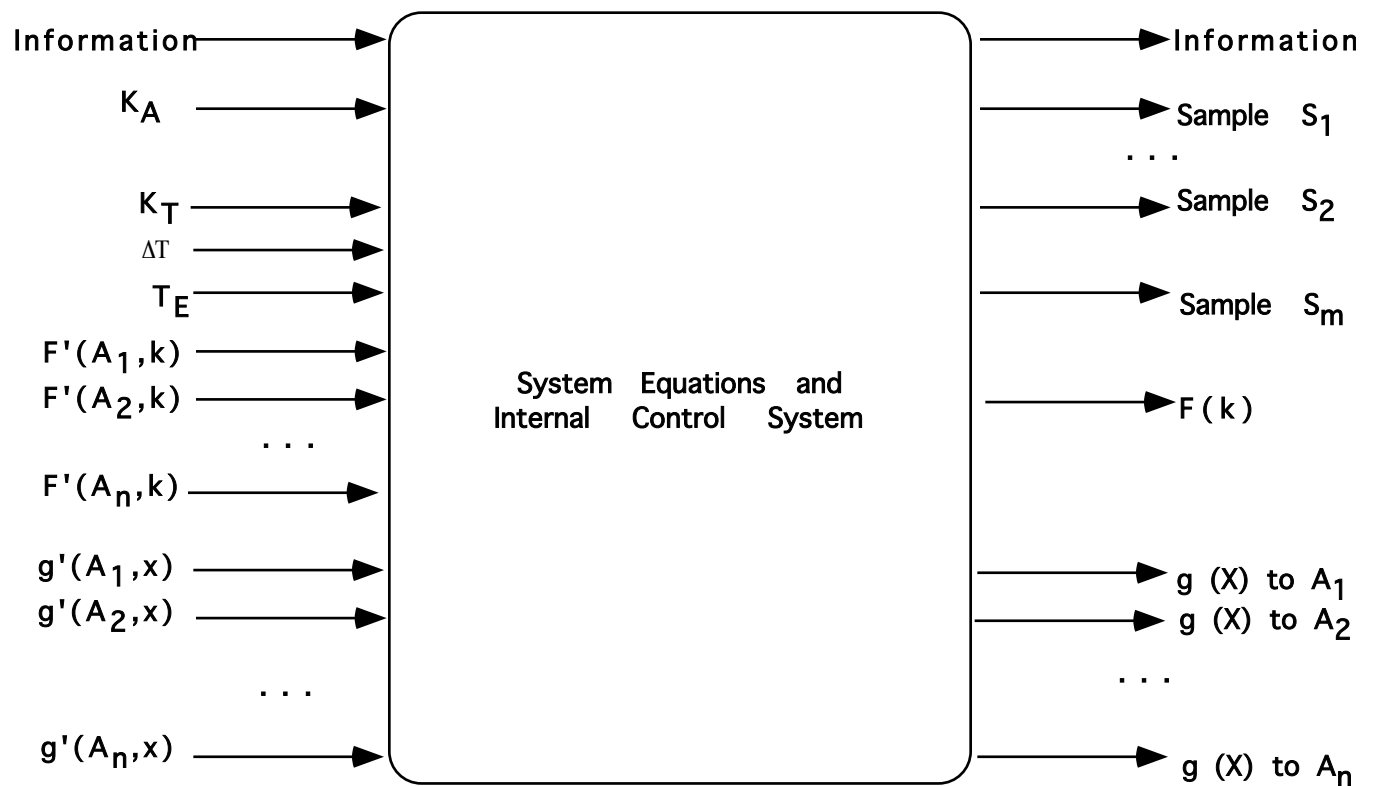
Figure 2. Control System Model for an Agent

The following are the equations that compute the probability tables for an agent. The probability values are limited to the range [0.01,1.0]; if a probability were to drop to 0 a link would be completely cut-off and never reactivated. *F'(Ai,k)* are the *F(k)* values fed back to the agent by its neighbors, and *g'(a,X)* is the usefulness of information fed back from a neighboring agent.

$$= 1.0 \qquad \text{for } 1.00 < F(k)g(X)$$

$$P_A(a,c) \quad\quad\quad\quad\quad = F(k)g(X) \quad\quad \text{for } 0.01 \le F(k)g(X) \le 1.00$$
$$= 0.01 \quad\quad\quad\quad \text{for } F(k)g(X) < 0.01$$

$$\quad\quad\quad\quad\quad\quad = 1.0 \quad\quad\quad\quad \text{for } 1.00 < F(k)h(Y)$$
$$P_S(s,c) \quad\quad\quad\quad\quad = F(k)h(Y) \quad\quad \text{for } 0.01 \le F(k)h(Y) \le 1.00$$
$$= 0.01 \quad\quad\quad\quad \text{for } F(k)h(Y) < 0.01$$

$$\quad\quad\quad\quad\quad\quad = 1.0 \quad\quad\quad\quad\quad\quad \text{for } 1.00 < F'(a,k)g'(a,X)$$
$$P_T(a,c) \quad\quad\quad\quad\quad = F'(a,k)g'(a,X) \quad \text{for } 0.01 \le F'(a,k)g'(a,X) \le 1.00$$
$$= 0.01 \quad\quad\quad\quad\quad\quad \text{for } F'(a,k)g'(a,X) < 0.01$$

F(k)	Scale factor that is generated by the agent's internal control system.
F'(a,k)	Scale factor that was fed back from a neighboring agent **a**.
g(X)	Usefulness of a information **X** that was received from agent **a** and classified into class **c**.
g'(a,X)	Usefulness of information **X** that was fed back from a neighboring agent **a**.
h(Y)	Usefulness of information **Y** that was received from sensor **s** and classified into class **c**.

Figure 3 shows the internal control system of an agent. This control system monitors the work load of agent. The scale factor $F(k)$ is adjusted until the expected amount of idle time T_E matches the measured amount of idle time $T_I(k)$. $F(k)$ directly impacts $T_I(k)$ because $F(k)$ is used to compute the probability tables that determine how information is routed and sampled. The control system is sampled every ΔT seconds. K_A and K_T are the parameters to the derivative feedback control system that determine how quickly and accurately an agent can adapt to changes in the multiagent system. The optimal values for these parameters depend on the dynamics of an actual system and have to be determined experimentally. The value $e(k)$ is the error term in the derivative feedback control system. The scale factor $F(k)$ is calculated by summing $e(k)$ over all sample periods. Derivative feedback is used because it has low overhead, and it provides better performance than a simple feedback control system.
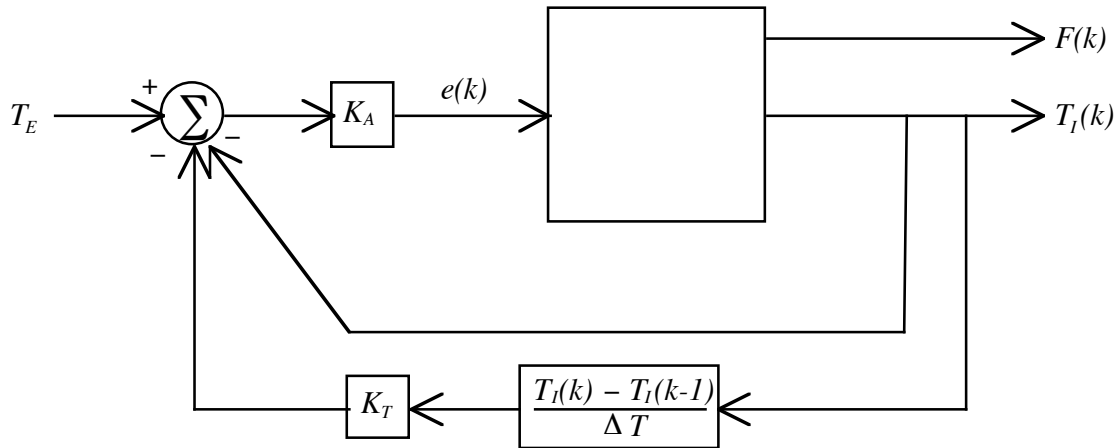
$$F(k) = \sum_{i=0}^{k} e(i)$$

$T_I(k)$ = measured amount of idle time in T seconds.

Figure 3. Internal Control System of an Agent

4.7. The ACCS for a Network of Rule-Based Agents

The reason that *g(X)* and *h(Y)* could not be defined earlier is that these usefulness equations depend on the characteristics of the problem solving component of an agent. Assuming a network of heterogeneous agents, each agent would have to have its own definition of fact usefulness; a scheduling agent may assign usefulness to classes of information in a manner different from a qualitative simulation agent. In our work we have elected to concentrate on rule-based agents, for reasons we will discuss in the following. This does not constrain the applicability of our work, since, given definitions for information usefulness the ACCS is applicable to any set of agents.

To define *g(X)* and *h(Y)*, the usefulness of a single fact must be defined. For rule based systems, the usefulness of a single fact is defined to be the percentage of rules in the rule base that a single fact *may potentially trigger*. In our prototype the rule based systems used were represented in a RETE network [Forgy 1982], and the rules that a fact may potentially trigger can be computed by the RETE nodes that are activated by this piece of information.

Once the usefulness of a single fact is defined, the usefulness of all the facts in a class is combined to determine the usefulness of an entire class of facts. The following two equations use the concepts of the classification of information, the usefulness of information, agent selfishness, and the cost of information to determine the usefulness of an entire class of facts. These equations assume that facts are classified and separated according to which agent transmitted the facts. Instead of using the *g(X)*, $U_A(a,c)$ is used, and instead of using *h(Y)*, $U_S(s,c)$ is used. The following is the definition of class usefulness along with the definitions of all the parameters that are used by these equations:

$$g(X) = U_A(a,c) = S \frac{1}{N_A(a,c)} \sum_{i=1}^{N_A(a,c)} \frac{U_{AF}(a,c,i)}{C_{AF}(a,c,i)} + (1-S)\frac{1}{n-1} \sum_{i=1, l \neq a}^{n} U_T(i,c)$$

$$h(Y) = U_S(s,c) = S \frac{1}{N_S(s,c)} \sum_{i=1}^{N_S(s,c)} \frac{U_{SF}(s,c,i)}{C_{SF}(s,c,i)} + (1-S)\frac{1}{n-1} \sum_{i=1}^{n} U_T(i,c)$$

$U_A(a,c)$ The usefulness of facts in class *c* received from agent *a*. This is the same as *g(X)* where *X* is the set of facts in class *c* received from agent *a*.

$U_S(a,c)$ The usefulness of facts in class *c* received from sensor *s*. This is the same as *h(Y)* where *Y* is the set of facts in class *c* received from sensor *s*.

S	A value between 0 and 1 that represents the selfishness of an agent. A value closer to 0 means that an agent is more concerned with the information needs of neighboring agent than with its own information needs. A value closer to 1 means that an agent is mainly concerned with its own information needs.
$N_A(a,c)$	The number of facts in class c received from agent a.
$N_S(s,c)$	The number of facts in class c received from sensor s.
$U_{AF}(a,c,i)$	The usefulness of the ith fact in class c received from agent a.
$C_{AF}(a,c,i)$	The cost of the ith fact in class c received from agent a.
$U_{SF}(s,c,i)$	The usefulness of the ith fact in class c received from sensor s.
$C_{SF}(s,c,i)$	The cost of the ith fact in class c received from sensor s.
n	The number of neighboring agents.
$U_T(i,c)$	The usefulness of facts in class c to the neighboring agent i. This is the same as $g'(X)$ where X is the set of facts in class c that were sent to agent i.

Both equations use the selfishness factor S. Notice that the factor S is used on the left term, and the factor $(1-S)$ is used on the right term. The left term is the usefulness of a class to the agent, and the right term is the usefulness of the class to the neighboring agents. When $S=0$, the left term is zero, and the usefulness of the class to an agent depends on the usefulness of this class to an agent's neighbors. Therefore, the agent is completely selfless. When $S=1$, the right term is zero, and the usefulness of the class to an agent depends solely on its own knowledge needs; therefore, the agent is completely selfish. When S has a value between 0 and 1, the left and right terms are weighted so an agent can take into account its own information needs with the left term and the information needs of its neighbors with the right term.

The left term computes the average usefulness of a set of facts that are members of a specific class. $U_{AF}(a,c,i)$, $C_{AF}(a,c,i)$, $U_{SF}(s,c,i)$, and $C_{SF}(s,c,i)$ are the usefulness and cost information on recently received facts from neighboring agents and sensors. The usefulness of each fact in the set is divided by the cost of the fact. This reduces the usefulness of expensive information and allows an agent to show a preference for the less expensive information routing paths. Only the most recently received facts in a class are used, $N_A(a,c)$, the number of facts in class c received from agent a, and $N_S(s,c)$, the number of facts in class c received from sensor s. Older facts are ignored because the system of agents is very dynamic and the communication strategy is constantly changing and it only makes sense to use facts that have been transmitted due to the use of the most recent communications strategy. This way, the communications strategy itself can be thought of as a state variable in a control system. All the agents work together to build a global communications strategy based on local viewpoints. Then, every sampling period, a new communications strategy replaces the old communications strategy. This new communications strategy is derived from the previous one based on the performance of the system over the sampling period. Therefore, only recently transmitted facts should be used to determine the usefulness of a class.

The right term computes the average usefulness of a class of facts to the neighboring agents. This value provides an agent with the information needs of its neighbors. Even though an agent's own problem solving component may not need a specific class of information, this agent may be required to route the information to an agent that does need that specific information class. This is the mechanism that provides an agent with the ability to perform information routing.

4.8. Model for an Adaptive Communications Strategy for Rule-Based Agents

Figure 1 shows the layered architecture for an agent with a rule based problem solving component. The Adaptive Communications Control System uses three internal models to make information routing and sensor sampling decisions. The sensor model is the probability table $P_S(s,c)$, the agent model is the probability table $P_T(a,c)$, and the model of itself is the probability table $P_A(a,c)$. A_i represents a connection to a neighboring agent, and S_j represents a connection to a neighboring sensor. Figure 4 shows the final version of the control system that the Adaptive Communications Control System uses to build and maintain its models of the world. The following system equations describe the relationships between the inputs and outputs of the control system. These system equations also calculate the probability tables that are used to make the information routing and sensor sampling decisions:

$$F(k) = F(k-1) + K_A \left(T_I(k) - T_E - K_T \frac{T_I(k) - T_I(k-1)}{\Delta T} \right)$$

$$P_S(s,c) \quad
\begin{aligned}
&= 1.0 &&\text{for } 1.00 < F(k)U_S(s,c) \\
&= F(k)U_S(s,c) &&\text{for } 0.01 \le F(k)U_S(s,c) \le 1.00 \\
&= 0.01 &&\text{for } F(k)U_S(s,c) < 0.01
\end{aligned}$$

$$P_T(a,c) \quad
\begin{aligned}
&= 1.0 &&\text{for } 1.00 < F'(a,k)U_T(a,c) \\
&= F'(a,k)U_T(a,c) &&\text{for } 0.01 \le F'(a,k)U_T(a,c) \le 1.00 \\
&= 0.01 &&\text{for } F'(a,k)U_T(a,c) < 0.01
\end{aligned}$$

The agent architecture and control system described above are designed to provide an agent with the seven behaviors that were defined in section 3.2:
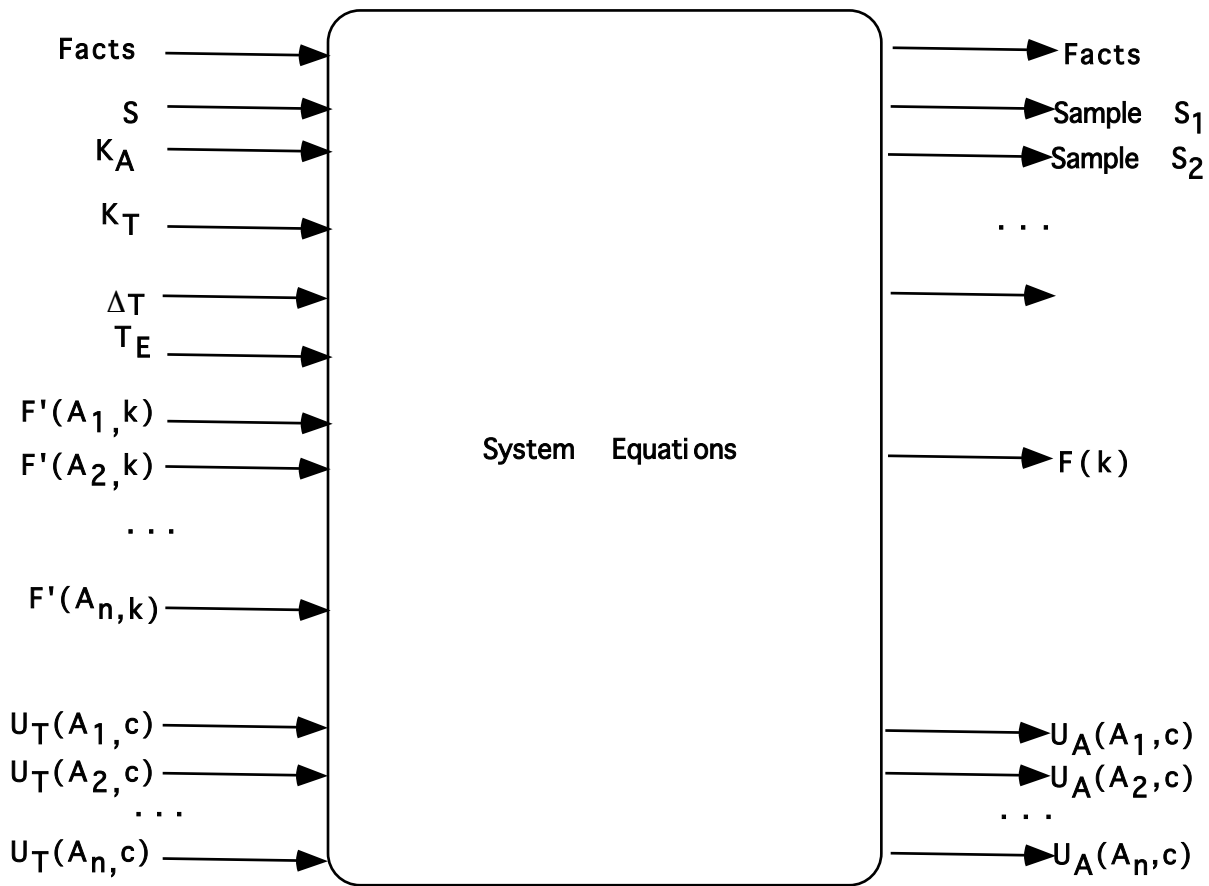
Figure 4. Control System Model for an Agent with a Rule Based System

1) Collect information that effects your own problem solving activities.

   This behavior is achieved with the selfishness factor and the equation for $U_A(a,c)$. If the selfishness factor is greater than zero, then the $U_A(a,c)$ values contain information on the agent's own information needs. These values are transmitted to neighboring agents and become $U_T(a,c)$ values that are used to make information routing decisions. When a specific class from a neighboring agent is found to be useful, the $U_A(a,c)$ value will be large, and thus the $U_T(a,c)$ value in the neighboring agent will also be large. This $U_T(a,c)$ value is used to compute $P_T(a,c)$. When the neighboring agent receives facts in the class c, they will likely be sent to the agent that found the class c useful. Therefore, by feeding back the usefulness of classes to neighboring agents, an agent insures that useful facts are sent its way.

2) Collect information that might effect the problem solving activities of your neighbors.

   This behavior is also achieved with the selfishness factor and the equation for $U_A(a,c)$. If the selfishness factor is less than one, then the $U_A(a,c)$ value contains information on the information needs of neighboring agents. These values are transmitted to neighboring agents, and are used to calculate $P_T(a,c)$ within the neighboring agents. When a neighbor

finds a class of facts useful, this increases the $U_T(a,c)$ values for that class. This in turn increases $U_A(a,c)$ which is fed back to other neighbors. This increases the probability that facts in class c are transmitted from these other neighbors. So, by feeding back the usefulness of classes to neighboring agents, the recursive nature of the class usefulness equations insures that agents collect information that is useful to their neighbors.

3) Collect the least expensive information possible.
   This behavior is addressed with the equations for $U_A(a,c)$ and $U_S(s,c)$. Since these equations divide the usefulness of individual facts by their cost, expensive facts are less useful than inexpensive facts. Since this information is fed back to neighboring agents, the less expensive sources of facts receive high class usefulness values. These classes have a higher probability of being sent, and thus the less expensive paths are favored.

4) Share information that might effect the problem solving activities of your neighbors with your neighbors.
   This behavior is satisfied through the use of the $P_T(a,c)$ tables. Whenever a fact is received, the $P_T(a,c)$ table is checked. If the fact is a member of a class that is useful to a neighboring agent, and the route that the fact has traveled is not more expensive than other paths, then the fact is likely transmitted to the neighboring agent.

5) Share your information needs with your neighbors.
   This behavior is satisfied by the sending of $U_A(a,c)$ values to neighboring agents. These values tell the neighboring agent which classes of facts are useful and which classes of facts are not useful.

6) Share your current work load with your neighbors.
   This behavior is satisfied by the computing the $F(k)$ values and sending them to neighboring agents. $F(k)$ values are computed from the amount of time that an agent is idle compared to the amount of time that an agent is expected to be idle. The scale factor $F(k)$ is adjusted in an attempt to match these two values. Therefore, $F(k)$ is an indirect indication of an agent's work load.

7) Budget time between problem solving, communicating with neighboring agents, and collecting information from sensors based on your own work load and the work load of neighboring agents.
   An agent must perform all problem solving possible, and an agent must handle all the messages that it has received. In addition, the probability table $P_S(s,c)$ must be monitored to determine the frequency that each neighboring sensor is sampled. The $F(k)$ and $U_T(a,c)$ values received from neighboring agents determine the importance of information routing. The facts received from sensors, an agent's internal $F(k)$ value, and the $U_T(a,c)$ values received from neighboring agents determine the importance of sensor sampling. Since the internal control system matches the measured idle time to the expected idle time, an agent will have time to perform all of its problem solving along with its information routing and sensor sampling activities. So, the relative weight of the

$P_T(a,c)$ table, the $P_S(a,c)$ table, and T_E determines how an agent budgets its time towards each of its activities.

5. Testbed and Experiments

5.1. Introduction

We developed a testbed to implement and evaluate the ACCS. The testbed operates on one or more hosts and communicates across a live network of SUN workstations. The layered agent architecture provided us with a natural division for the implementation into a set of modules.

The low level communications module is implemented using the Remote Procedure Call (RPC) protocol that is supported under SunOS. The details of the communications module's implementation can be found in [Kinney 1992a]. For the rule-based problem solving component of an agent we used CLIPS, a language that allows the creation of (primarily) forward reasoning rule-based systems [NASA COSMIC 1989]. We selected CLIPS as our problem solving language because we had access to the source code and the RETE network, and because CLIPS programs can be easily embedded inside other applications. To make prototyping simple we defined a language that allows the definition of a network of agents, and also an extension to CLIPS which allows the passing of messages. Examples can be found in [Kinney 1992b].

Several experiments were performed to test the performance of the ACCS. These include experiments on fully connected networks with varying populations of agents, and non fully connected networks with varying populations and topologies. The discussion of each experiment contains a figure showing the topology of the experiment along with a condensed version of each rule base. The following symbols are used to represent agents and sensors:
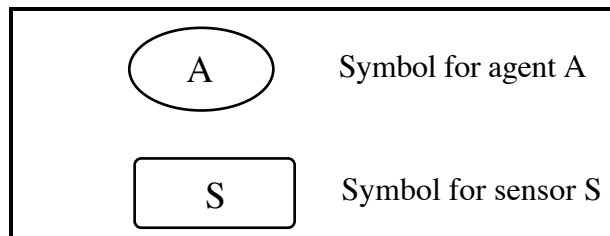

Figure 5: Symbols used to represent agents and sensors

Communication links between agents and sensors are represented by lines connecting the different nodes. If a communication link is labeled with a value, that value represents the cost of sending a message across that link. If a link is not labeled, then the cost is assumed to be 0.0.

Since we concentrated on studying the experimental performance of our model, we did not use any real domain knowledge or application. Our systems contained randomly generated rule-bases and sensors. The goal was to run simulations of various agent configurations and tasks, and this required domain-independent simulations. A condensed version of the rule base is listed under each agent. These rules show the relationship between different classes of information.

For example, the rule v5→v4, means that facts in class v5 generate facts in class v4. A single class is also listed under each sensor. This shows the class of facts that a sensor will generate when it is sampled. For each experiment, there is a class of facts that is considered a final conclusion. Whenever a system generates a fact in this special class, the system has successfully performed some amount of work. The efficiency of a multiagent system is measured by determining the rate at which final conclusions are generated, and how many resources the system consumed in order to generate those final conclusions. An adaptive communications strategy is effective if the efficiency of the multi-system improves over time.

5.2. Performance Measurements

Each agent process collects performance measurements as an experiment is performed. These measurements include the resources that an agent uses and many of the state variables contained within an agent's system equations. The resources monitored include the number of messages an agent sends to neighboring processes, the number of rules an agent fires, the number of final conclusions an agent generates, and the total cost associated with an agent using these resources. The state variables monitored include the agent's work load (T_I), the probability that neighboring sensors are sampled ($P_S(s,c)$), and the probability that facts in a specific class are transmitted to neighboring agents ($P_T(a,c)$).

The collection of these measurements is coordinated by the user interface process. The user is allowed to set the length of time the experiment is to be run, and how often the measurements are taken. The user interface process sends a control message to the agent processes each sampling period. The agent processes respond to this message by writing all their performance measurements to an output file. At the end of the experiment all of the output files are processed to generate agent level and system level performance statistics. The following is the list of statistics generated:

System Level Performance Statistics:
> **Final Conclusions per Second** : Shows the rate at which final conclusions are generated. It is computed by summing the final conclusion generation rates of all the agents in the system.
> **Rule Firings per Second** : Shows the rate at which rules are fired in a multiagent system. It is computed by summing the rule firing rates of all the agents in the system.
> **Messages per Second** : Shows the rate at which messages are transmitted in a multiagent system. It is computed by summing the message transmission rates of all the agents in the system.
> **Cost per Second** : This is a measure of the resources the multiagent system consumes in a time interval. Both sending messages and firing rules consume resources. Communications links have an associated value that represents the cost of sending a message across the link. Agents also have an associated value that represents the cost of firing a rule. This measurement shows the total cost of transmitting messages and firing rules. It is computed by summing the cost rates of all the agents in the system.

Rule Firings per Final Conclusion : Shows the number of rules that are fired for each final conclusion generated. It measures how efficiently the agents use their rule firing resources to generate final conclusions. This statistic is significant because a multiagent system can be analyzed to determine its optimal value.

Messages per Final Conclusion : Shows the number of messages that are transmitted for each final conclusion generated. It measures how efficiently the agents use their communications resources to generate final conclusions. This statistic is significant because a multiagent system can be analyzed to determine its optimal value.

Cost per Final Conclusion : Shows of the resources the multiagent system consumes in generating one final conclusion. It is computed by dividing the Cost per Second statistic by the Final Conclusions per Second statistic.

Agent Level Performance Statistics:

Rule Firings per Second : Shows the rate at which an agent is firing rules.

Messages per Second : Shows the rate at which an agent is transmitting messages. These include fact messages, feedback messages, and synchronization messages.

Final Conclusions per Second : Shows the rate at which an agent is generating final conclusions.

Cost per Second : Shows the rate at which an agent is consuming resources.

Cost per Rule Firing : Shows how many resources an agent consumes to fire a rule.

Messages per Rule Firing : Statistic shows how many messages an agent transmits to fire a rule.

T_I : Shows the amount of time an agent is idle in a given time period.

$P_S(s,c)$: Shows the probability that neighboring sensors will be sampled.

$P_T(a,c)$: Shows the probability that facts in class c will be transmitted to agent a.

5.3. Experiments with Fully Connected Networks

The first four experiments involve fully connected networks of agents with different populations. These populations range from two in the first experiment to five in the fourth experiment. Even though the agents form a fully connected network, the sensors are not fully connected. Instead, a sensor is connected to one of the agents in the system. Each experiment also involves different rule bases. This is done in an attempt to eliminate the dependence of the ACCS on certain rule base organizations. The first four experiments make several assumptions to prove that a simplified version of the adaptive communications strategy is effective:

1) The cost of transmitting a message across a link is assumed to be zero.
2) The cost of firing a rule in an agent is assumed to be zero.
3) The agents are assumed to be fully connected. With this organization, the furthest a fact has to be routed is through one agent node. Since agents don't have to act as information routers, an agent can be assumed to be completely selfish. Therefore, an agent's selfishness factor is assumed to be 1.0. This eliminates the right factor of the class usefulness equations.

With these assumptions, the system equations reduce to the following:

$$F(k) = F(k-1) + K_A \left(T_I(k) - T_E - K_T \frac{T_I(k) - T_I(k-1)}{\Delta T} \right)$$

$$U_A(a, c) = \frac{1}{N_A(a, c)} \sum_{i=1}^{N_A(a,c)} U_{AF}(a, c, i)$$

$$U_S(s, c) = \frac{1}{N_S(s, c)} \sum_{i=1}^{N_S(s,c)} U_{SF}(s, c, i)$$

$$P_S(s,c) \quad \begin{aligned} &= 1.0 & &\text{for } 1.00 < F(k)U_S(s,c) \\ &= F(k)U_S(s,c) & &\text{for } 0.01 \le F(k)U_S(s,c) \le 1.00 \\ &= 0.01 & &\text{for } F(k)U_S(s,c) < 0.01 \end{aligned}$$

$$P_T(a,c) \quad \begin{aligned} &= 1.0 & &\text{for } 1.00 < F'(a,k)U_T(a,c) \\ &= F'(a,k)U_T(a,c) & &\text{for } 0.01 \le F'(a,k)U_T(a,c) \le 1.00 \\ &= 0.01 & &\text{for } F'(a,k)U_T(a,c) < 0.01 \end{aligned}$$

Each experiment was run for a period of 60 seconds, and performance measurements were taken once a second. The table for each experiment shows how efficiently the system performed. These statistics show the system's average performance over the last twenty seconds of the experiment. These measurements were taken at the end of the experiment so the system's performance is measured while it was in steady state. The column labeled "Initial" shows the performance of the system without an adaptive communications strategy. Instead, each agent in the system sends all new facts to all neighboring agents. The column labeled "ACCS" shows the performance of the adaptive communications strategy. The system started with the same strategy as the "Initial" column, but the adaptive communications control system was allowed to modify the strategy in an attempt to improve the system's efficiency. The last column contains the optimal performance statistics which were determined by examining the entire multiagent system: the minimal path from sensor input to final conclusion generation was identified, and the number of messages passed and rules fired along this path were counted. The following three graphs show the main performance statistics for the first experiment. The figures and tables following these graphs show the network topology, rule bases, and system wide performance statistics for the first four experiments.
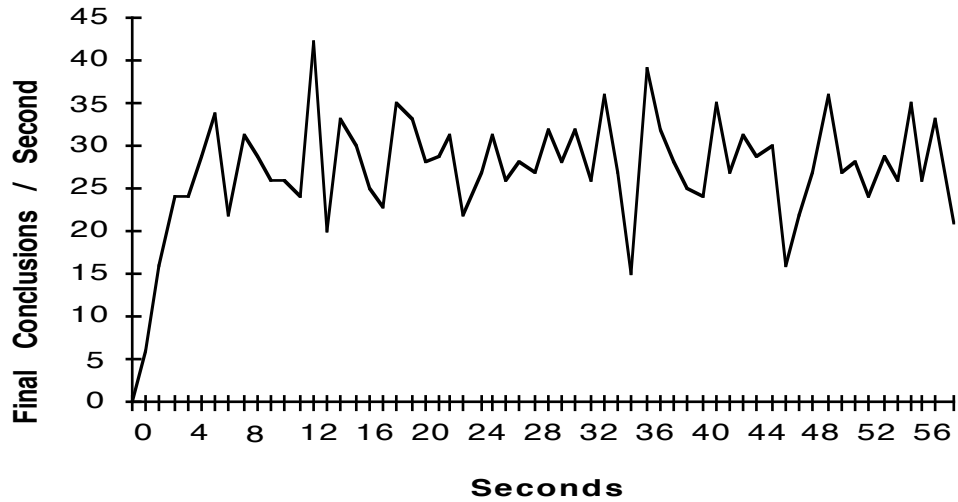
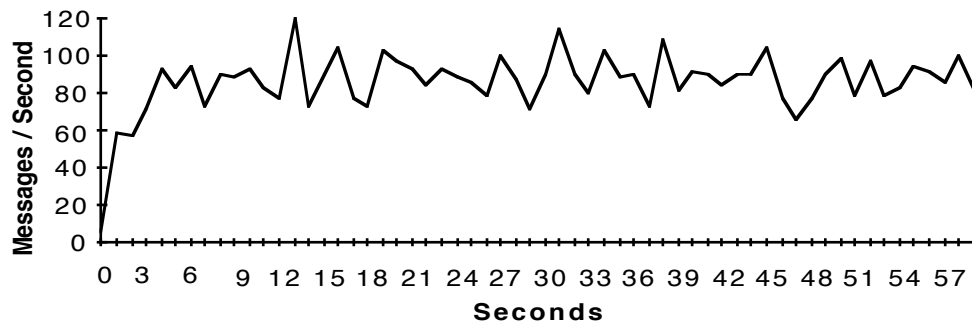**Figure 6:** Final conclusions per second for Experiment 1

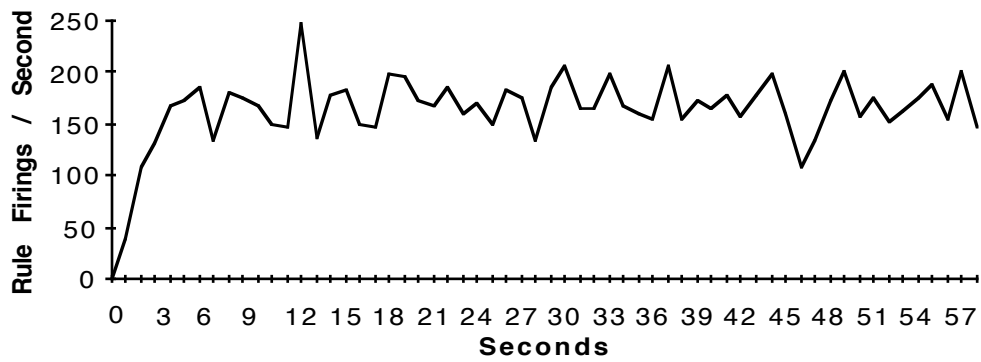**Figure 7:** Messages per second for Experiment 1

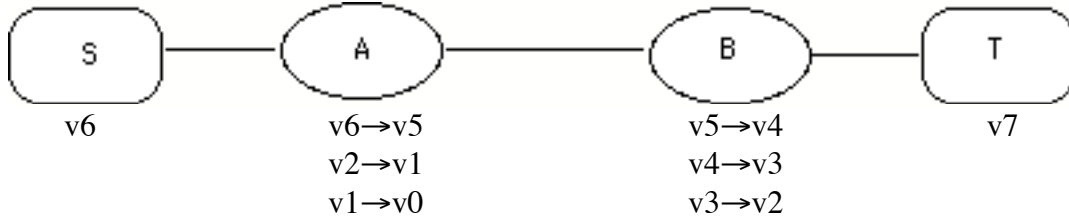**Figure 8:** Rule firings per second for Experiment 1

Figure 9. Network topology of Experiment 1

	Initial	ACCS	Optimal
Final Conclusions / Second	22.90	27.55	---
Messages / Second	232.65	88.05	---
Rule Firings / Second	138.70	167.0	---
Messages / Final Conclusion	10.16	3.20	3
Rule Firings / Final Conclusion	6.05	6.06	6

Table 1. Summary of results for Experiment 1
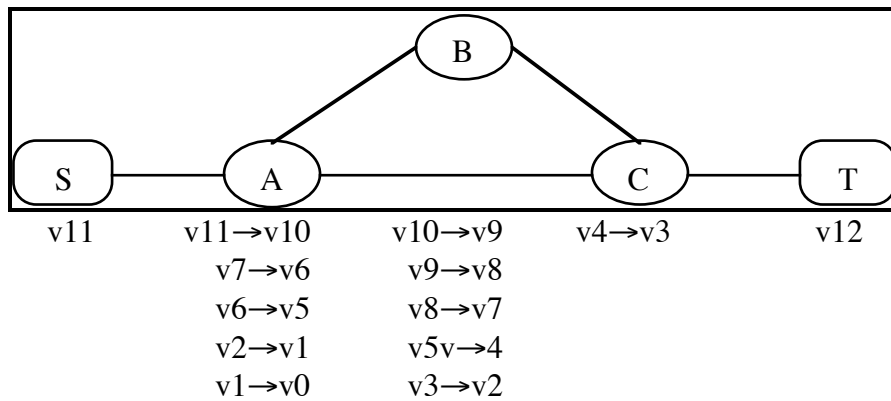
Figure 10. Network topology of Experiment 2

	Initial	ACCS	Optimal
Final Conclusions / Second	2.80	18.95	---
Messages / Second	395.40	145.45	---
Rule Firings / Second	112.45	212.40	---
Messages / Final Conclusion	141.21	7.68	7
Rule Firings / Final Conclusion	40.16	11.21	11

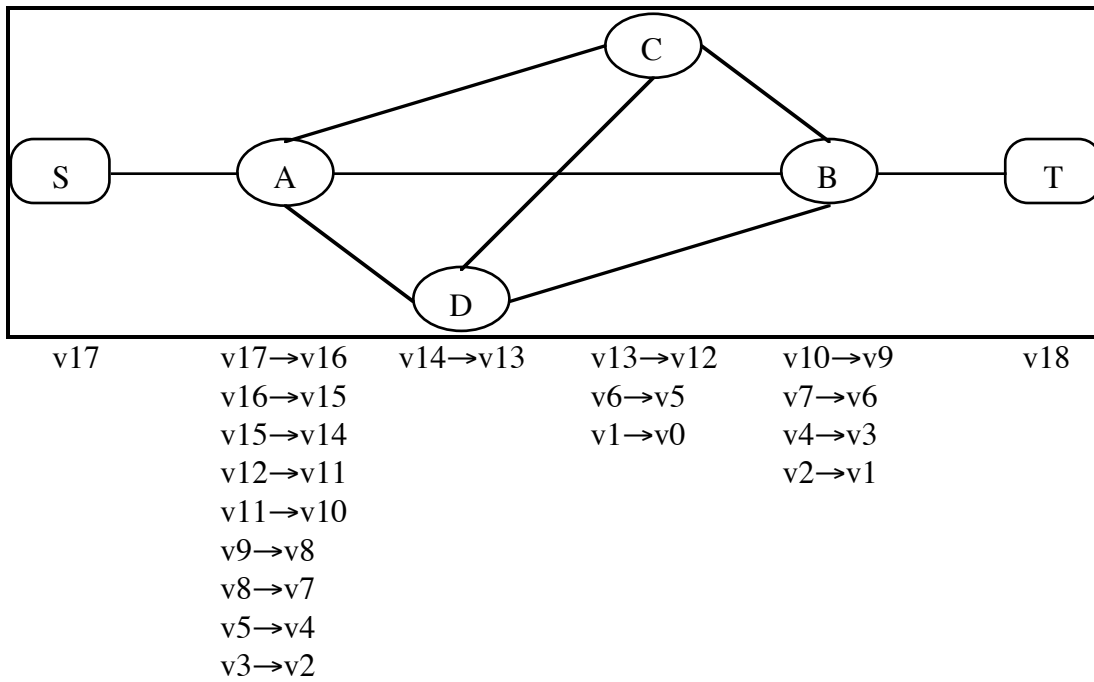**Table 2.** Summary of results for Experiment 2

v17	v17→v16	v14→v13	v13→v12	v10→v9	v18
	v16→v15		v6→v5	v7→v6	
	v15→v14		v1→v0	v4→v3	
	v12→v11			v2→v1	
	v11→v10				
	v9→v8				
	v8→v7				
	v5→v4				
	v3→v2				

Figure 11. Network topology of Experiment 3

	Initial	ACCS	Optimal
Final Conclusions / Second	0.65	11.75	---
Messages / Second	580.60	192.95	---
Rule Firings / Second	70.65	220.05	---
Messages / Final Conclusion	893.23	16.42	13
Rule Firings / Final Conclusion	108.69	18.73	17

Table 3. Summary of results for Experiment 3
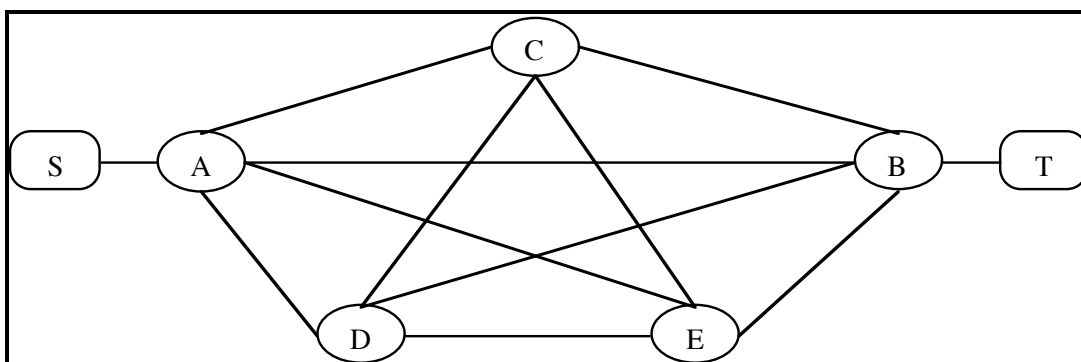
v5   v5→v4   v2→v1   v3→v2   v1→v0   v4v→v3   v6

Figure 12. Network topology of Experiment 4

	Initial	ACCS	Optimal
Final Conclusions / Second	0.85	43.70	---
Messages / Second	311.40	266.80	---
Rule Firings / Second	16.50	231.65	---
Messages / Final Conclusion	366.35	6.11	5
Rule Firings / Final Conclusion	19.41	5.30	5

Table 4. Summary of results for Experiment 4

All of these tables show that the adaptive communications strategy increased the rate at which final conclusions were generated, decreased the message load, increased the rule firing rate, and almost reached the optimal values for the resources consumed per final conclusion generated. In fact, the difference between the adaptive communications strategy and these optimal values can be attributed to the overhead of the adaptive communications strategy and errors in the performance measurements.

Graph 4 shows the probability that Agent A will sample sensor S during the third experiment. This graph has a close correlation to the work loads tracked during the experiment: the sharp rise in the sensor sampling probability corresponds to the point that Agent A had a large amount of idle time. The internal control system increased $F(k)$, which in turn increase $P_S(s,c)$.
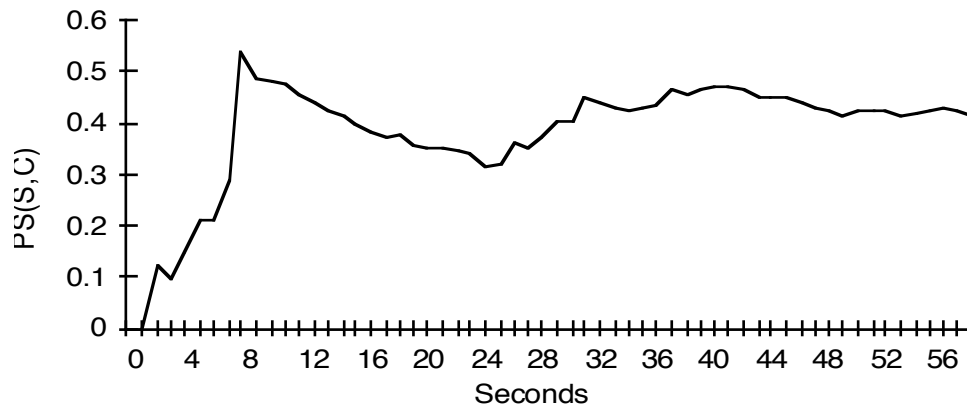

Figure 13. Sensor sampling probability for Agent A

5.4. An Experiment with Distracting Rule Sets

This experiment uses a non-fully connected network of agents, and a selfishness factor of 0.8 The means that an agent is mainly concerned with its own information needs, and a little concerned with the information needs of its neighbors. This experiment also contains a subset of rules that do not move the system towards generating a final conclusion. These rules are placed in the system to measure the effects of distracting rules on the performance of the system. The adaptive communications strategy improved the performance of the system across all the system wide performance measures, but the results for this system are not as good as for the fully connected networks. This is due to the fact that Agent B must route information between Agent A and Agent C, and Agent B contains a set of distracting rules that degrade the system's performance.

The only unusual statistic is the number of rule firings per final conclusion while using the initial communications strategy. This value of approximately 5 is less than the minimum expected value of 8. The only reasonable explanation, though it has not been verified, is that Agent B fell behind in firing rules as it was overwhelmed with work to perform. The first few of its rules would get fired, and the facts generated would be sent to neighboring agents. However, Agent B could not finish with all the rule firings and message transmissions before more facts would be sent from the neighboring agents who had much smaller work loads. Agent A had to keep increasing the sensor sampling rate for S in order to keep Agent A and Agent C busy. This generated too much work for Agent B to perform. Agent B fell behind, and on average, never fired half of its rules.
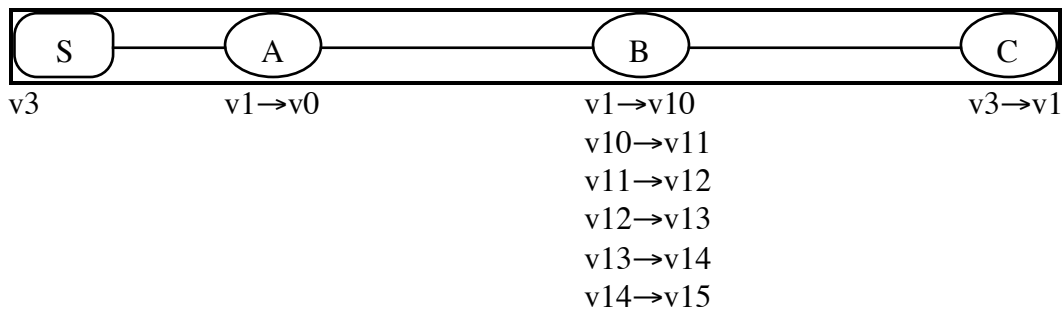
S	A	B	C
v3	v1→v0	v1→v10	v3→v1
		v10→v11	
		v11→v12	
		v12→v13	
		v13→v14	
		v14→v15	

Figure 14. Network topology of Experiment 5

	Initial	ACCS	Optimal
Final Conclusions / Second	17.40	25.05	---
Messages / Second	250.40	131.30	---
Rule Firings / Second	92.25	199.80	---
Messages / Final Conclusion	14.39	5.24	5
Rule Firings / Final Conclusion	5.30	7.98	2

Table 5. Summary of results for Experiment 5

5.5. An Experiment with an Information Router

This experiments shows how the adaptive communications strategy works with a hierarchical topology. Agent A does not have any rules, and is the only connection to the other agents in the system. Agent A is at the top of the hierarchy, and it must assume the role of an information router and coordinate the activities of the remaining agents who are at the bottom of the hierarchy. All the system parameters that were experimentally determined from the previous examples are used in this system. The initial communications strategy performed very poorly in this experiment, and the adaptive communications strategy improved upon this initial strategy for all the system wide performance statistics.
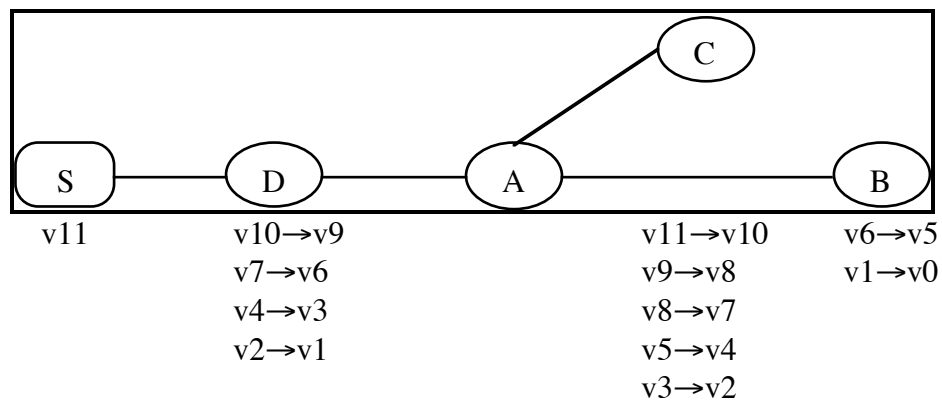

Figure 15. Network topology of Experiment 6

	Initial	ACCS	Optimal
Final Conclusions / Second	1.00	7.10	---
Messages / Second	349.75	230.25	---
Rule Firings / Second	105.80	136.50	---
Messages / Final Conclusion	349.75	32.43	21
Rule Firings / Final Conclusion	105.80	19.23	11

Table 6. Summary of results for Experiment 6

5.7. An Experiment with Resource Costs

This final experiment examines the performance of a system in which there is a cost associated with transmitting messages. Here the rule bases are very simple, and are purposely selected to see if the system can favor the least expensive path. Notice that the path through Agent C is much more expensive than the path through agent D. Ideally, Agent A should choose to send most of its facts in class v2 to agent D and not to Agent C.
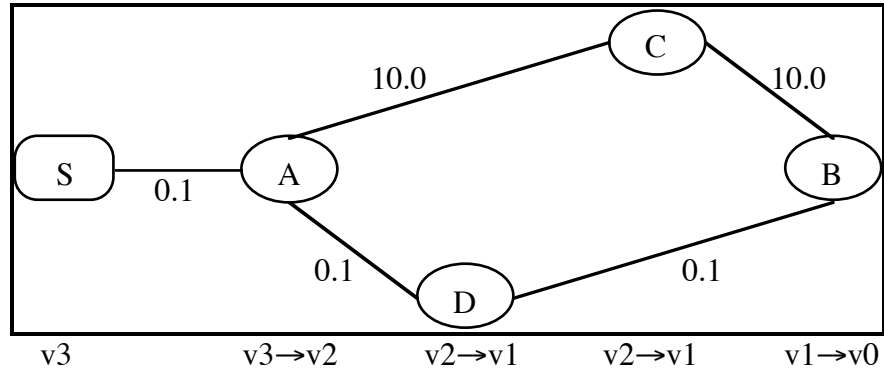
Figure 16. Network topology of Experiment 7

	Initial	ACCS	Optimal
Final Conclusions / Second	11.30	20.75	---
Messages / Second	501.50	137.50	---
Rule Firings / Second	91.85	106.95	---
Total Cost / Second	1045.98	439.33	---
Messages / Final Conclusion	44.38	6.63	3
Rule Firings / Final Conclusion	8.13	5.15	3
Cost / Final Conclusion	92.56	21.17	0.3

Table 7. Summary of results for Experiment 7

Once again, the adaptive communications strategy did much better than the initial communications strategy, but it did not get close to the optimal values. The worst statistic is the Cost per Final Conclusion.

The reason for the poor performance of this multiagent system can be seen in the probability that the classes of facts are sent from Agent A. Graph 5 shows the probability that facts are transmitted from Agent A to Agent C. This communications link should be almost completely cut off due to the high cost of transmitting across this link. Instead, Agent A has a relatively high probability that facts in class v2 are sent to Agent C. Graph 6 shows the real problem: the probability that different classes are sent from Agent A to Agent D has very large oscillations. This system, even though it successfully improved the performance, is out of control. The current model for an adaptive communications strategy does not handle cost information well, and as a result, the systems that include cost information may not achieve steady state. The basic problem is that the model divides the usefulness of a fact by its cost. When there are radically different cost values for two different paths, the class usefulness values can change by large amounts. This induces the oscillations whose amplitudes are too large for the control system. These oscillations in the probabilities are the cause of the high cost in the system.
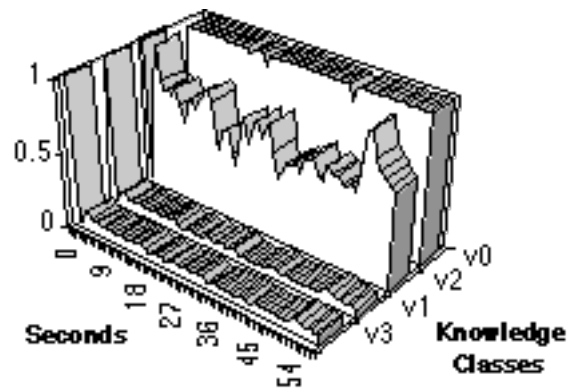
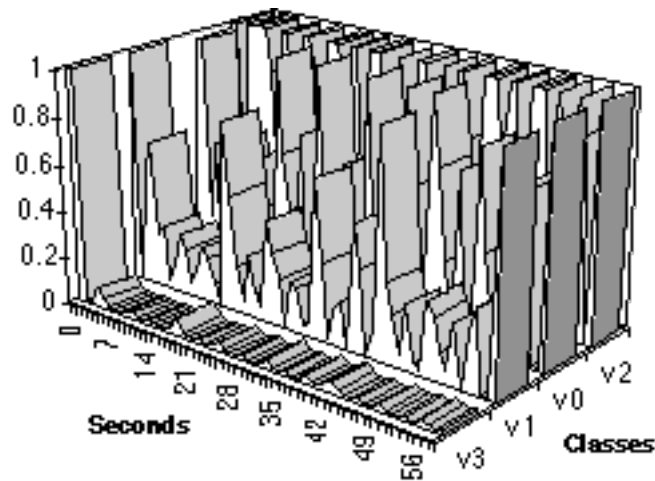**Figure 17:** Experiment 7: Probability of Agent A sending facts to Agent C

Figure 18: Experiment 7: Probability of Agent A sending facts to Agent D

5.7. Summary of Results

The ACCS improves the efficiency of fully connected networks, and larger agent populations result in greater improvements in efficiency. This not only shows the effectiveness of the adaptive communications strategy, but also the ineffectiveness of the initial communications strategy. In general, the adaptive communications strategy improved the performance of all the experiments run. The best improvement was seen for fully connected networks with no cost information. When the assumption of a fully connected network of agents is removed, the adaptive communications strategy is still very good, but it falls short of the optimal values. Finally, when cost information is introduced, the adaptive communications strategy still provides an improvement over the initial communications strategy, but it is even farther from the optimal performance levels.

The last three experiments show where the adaptive communications strategy needs improvement. It cannot reach optimal performance levels in systems that contain agents with distracting rule sets. It also has problems in systems where agents are required to specialize as information routers. The adaptive communications strategy showed its worst performance in systems where cost information is added.

6. Conclusions

For an agent in a multiagent system to contribute to the problem solving group, it must exhibit behaviors that help the group move towards the solution to a given problem. The largest obstacle that an agent must overcome is how is decides to communicate with other agents in the system. The behaviors that provide an agent with the ability to learn how to communicate with other agents were identified. From these behaviors, a control system for an agent was derived. This control system monitors the information entering and leaving an agent and determines what types of information are important to an agent. Each agent examines its own information needs and the information needs of its neighbors to determine an agent's own communications strategy. When the agents work together to solve a problem, the combined affect of their local communications strategies form a global communications strategy that controls the flow of information throughout a multiagent system.

The theory for an adaptive communications strategy was developed and applied to agents that use a rule based system to perform their problem solving activities. The control system, which is the central component of an adaptive communications strategy, was refined to reflect the needs of rule based systems. Then, a testbed was designed and built to perform experiments on this adaptive communications strategy. This testbed was used in various experiments and it generated performance measurements that showed the performance of the adaptive communications strategy under different conditions. The results from these experiments showed that an adaptive communications strategy is very effective on fully connected networks of agents, greatly improves the performance of general topologies, but has problems with distracting rule sets, networks that contain nodes that are communication bottlenecks, and networks that have costs associated with links.

7. Acknowledgments

8. References

Bond, Alan H. and L. Gasser (Eds.). 1988. *Readings in Distributed Artificial Intelligence*, Morgan Kaufmann, San Mateo, CA, 3-35.

Brazdil, P., M. Games, S. Sian, L. Torgo, and W. van de Velde. 1991. "Learning in Distributed Systems and Multiagent Environments," *Proceedings of the European Working Session on Learning*, Porto, Portugal, 413-423.

Brazdil, P. and S. Muggleton. 1991. "Learning to Relate Terms in a Multiple Agent Environment," *Proceedings of the European Working Session on Learning*, Porto, Portugal, 424-439.

Bui, H.H., D. Kieronska and S. Venkatesh. 1996. "Learning Other Agents' Preferences in Multiagent Negotiation," *AAAI-96*, 114-119.

Cammarata S., D. McArthur and R. Steeb. 1983. "Strategies of Cooperation in Distributed Problem Solving," *IJCAI-83*, 767-770.

Carmel, D. and S. Markovitch. 1996. "Learning Models of Intelligent Agents," *AAAI-96*, 62-67.

Conry, S.E., R.A. Meyer and R. P. Pope. 1989. "Mechanisms for Assessing Nonlocal Impact of Local Decisions in Distributed Planning," in: *Distributed Artificial Intelligence*, L. Gasser and M. Huhns (Eds.), Morgan Kaufmann Publishers, 245-257.

Decker, Keith S. 1994. "Distributed Artificial Intelligence Testbeds," in: *Foucndations of Distributed Artificial Intelligence*, G. O'Hare and N. Jennings (Eds.), Wiley Inter-Science.

Decker, Keith S. and V.R. Lesser. 1994. "Analyzing the Need for Meta-Level Communication," *IJCAI-94*.

Decker, K.S. and V.R. Lesser. 1995. "Designing a Family of Coordination Algorithms," *Proc. First Int. Conf. on Multiagent Systems*, AAAI Press, 73-80.

Durfee, Edmund H. and Victor R. Lesser. 1987. "Using Partial Global Plans to Coordinate Distributed Problem Solvers," *IJCAI-87*, Milan, Italy, 875-883.

Durfee, Edmund H. and Victor R. Lesser. 1988. "Incremental Planning to Control a Time-Constrained, Blackboard-Based Problem Solver," *IEEE Transactions on Aerospace and Electronic Systems* 24(5), 647-662.

Durfee, Edmund H. and V.R. Lesser. 1989. "Negotiating Tasking Decomposition and Allocation Using Partial Global Planning," in: *Distributed Artificial Intelligence*, L. Gasser and M. Huhns (Eds.), Morgan Kaufmann Publishers, 229-243.

Durfee, Edmund H. and Victor R. Lesser. 1991. "Partial Global Planning: A Coordination Framework for Distributed Hypothesis Formation," *IEEE Transactions on Systems, Man, and Cybernetics* 21(5), 1167-1183.

Durfee, Edmund H., Victor R. Lesser, and Daniel D. Corkill. 1987. "Coherent Cooperation Among Communicating Problem Solvers," *IEEE Transactions on Computers*, Vol. C-36(11), 1275-1291.

Ensor, J. Robert and John D. Gabbe. 1985. "Transaction Blackboards," *IJCAI-85*, 340-344.

Erman, Lee D., Frederick A. Hayes-Roth , Victor R. Lesser and D. Raj Reddy. 1980. "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty," *Computing Surveys* 12(2), 213-253.

Fagin, Ronald and Moshe Y. Vardi. 1986. "Knowledge and Implicit Knowledge in a Distributed Environment: Preliminary Report," *Proceedings of the Conference on the Theoretical Aspects of Reasoning about Knowledge*, Joseph Y. Halpern (Ed.), 187-206.

Fenster, M. S. Kraus, and J.S. Rosenschein. 1995. "Coordination without Communication: Experimental Validation of Focal Point Techniques," *Proc. First Int. Conf. on Multiagent Systems*, AAAI Press, 102-108.

Foisel, R., V. Chevrier, and J-P Haton. 1996. "Improving Global Coherence by Adaptive Organization in a Multi-Agent System," *Proc. Second Int. Conf. on Multiagent Systems,* AAAI Press, 435.

Fox, Mark S. 1981. "An Organizational View of Distributed Systems," *IEEE Transactions on Systems, Man, and Cybernetics,* vol. SMC-11 (1), 70-80.

Gallant, S. 1988. "Connectionist Expert Systems," *Communications of ACM*, Vol. 31, 152-168.

Georgeff, Michael. 1983. "Communication and Interaction in Multiagent Planning," *Proc. of the 8th International Joint Conference on Artificial Intelligence*, Karlsruhe, Germany, 125-129.

Hayes-Roth, Barbara. 1985. "A Blackboard Architecture for Control," *Artificial Intelligence* 26, 251-321.

Hayes-Roth, Barbara. 1995. "An Architecture for Adaptive Intelligent Systems", *Artificial Intelligence* (72) 1-2, 329-365.

Jacobs, Walter. 1971. "A Structure for Systems that Plan Abstractly," *Proceedings of the AFIPS Conference, Spring Joint Computer Conference,* vol. 38, 357-364.

Kinney, Michael D. 1992a. *A Multi-Agent Communications Tool Using Remote Procedure Calls,* CECASE-TR-8640-12, Center for Excellence in Computer Aided Systems Engineering, The University of Kansas.

Kinney, Michael D. 1992b. *A Testbed for an Adaptive Communications Strategy for Multi-Agent Systems*, CECASE-TR-8640-27, Center for Excellence in Computer Aided Systems Engineering, The University of Kansas.

Kornfeld, W.A. 1979. "ETHER: A Parallel Problem Solving System," *IJCAI-79,* 490-492.

Lesser, Victor R. and Daniel D. Corkill. 1981. "Functionally Accurate, Cooperative Distributed Systems," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-11(1), 81-96.

Lesser, Victor R. and Lee D. Erman. 1980. "Distributed Interpretation: A Model and Experiment," *IEEE Transactions on Computers* C-29(12), 1144-1163.

Malone, Thomas W. 1987. "Modeling Coordination in Organizations and Markets," *Management Science*, Vol. 33(10), 1317-1332.

NASA COSMIC. 1989. *CLIPS Advanced Programming Guide,* Version 4.1.

Neiman, Daniel E., David W. Hildum and Victor R. Lesser. 1994. "Exploiting Meta-Level Information in a Distributed Scheduling System," *IJCAI-94*.

Ohkno, T., K. Hiraki, and Y. Anzai. 1996. "Reducing Communication Load on Contract Net by Case-Based Reasoning - Extension with Directed Contract and Forgetting," *Proc. Second Int. Conf. on Multiagent Systems*, AAAI Press, 244-251.

Prasad, M.V. Nagendra, V.R. Lesser, and S. Lander. 1996. "Learning Organizational Roles in a Heterogeneous Multi-agent System," *AAAI Symposium on Adaptation, Coevolution and Learning in Multiagent Systems*, AAAI Technical Report SS-96-01, 73-77.

Rosenschein, Jeffery S. 1982. "Synchronization of Multiagent Plans," *Proceedings of the National Conference of the American Association for Artificial Intelligence*, Pittsburgh, PA, 115-119.

Shahom, Yoav. 1993. "Agent-Oriented Programming," *Artificial Intelligence* 60, 51-92

Shoham, Yoav and M. Tennenholtz. 1995. "On Social Laws for Artificial Agent Societies: Off-Line Design", *Artificial Intelligence* (73) 1-2, 231-252.

Shaw, Michael J. and Andrew B. Whinston. 1990. "Learning and Adaptation in Distributed Artificial Intelligence Systems," in: L. Gasser and M.N. Huhns (Eds.), *Distributed Artificial Intelligence* Vol. II, Pitman Publishing/Morgan Kaufmann Publishers, 413-429.

Shoham Y. and M. Tennenholtz. 1992. "On the Synthesis of Useful Social Laws for Artificial Agent Societies (Preliminary Report)," *AAAI-92*, 704-709.

Smith, Reid G. and Randal Davis. 1981. "Frameworks for Cooperation in Distributed Problem Solving," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-11(1), 61-70.

Smith, Reid G. 1980. "The Contract Net Protocol : High Level Communication and Control in a Distributed Problem Solver," *IEEE Transactions on Computers*, Vol. C-29(12), 1104-1113.

Steeb, Randal, Stephanie Cammarata, Fredrick A. Hayes-Roth, Perry W. Thorndyke, Robert B. Wesson. 1981. *Architectures for Distributed Air-Traffic Control*, Technical Report R-2728-ARPA, 9-30.

Tsatsoulis, C. and G. Yee. 1996. "Learning Reliability Models of Other Agents in a Multiagent System," *AAAI Workshop on Intelligent Adaptive Agents*, AAAI Technical Report.

Turner, Elise H. 1994. "Exploiting Problem Solving to Select Information to Include Dialogues between Cooperating Agents," *Sixteenth Annual Conference of the Cognitive Science Society*, 882-886.

Wesson, Robert, Fredrick Hayes-Roth, John W. Burge, Cathleen Stasz, and Carl A. Sunshine. 1981. "Network Structures for Distributed Situation Assessment," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-11(1), 5-23.

Whitehair, R. and V.R. Lesser. 1993. *A Framework for the Analysis of Sophisticated Control in Interpretation Systems,* Computer Science Technical Report 93-53, University of Massachusetts.