Appeared in Proceedings of the Tenth International Symposium on Unmanned Untethered Submersible Technology, September 7–10, 1997, Durham NH. Published by Autonomous Undersea Systems Institute, 86 Old Concord Turnpike, Lee NH, 03824.

Cooperative Behavior in an Autonomous Oceanographic Sampling Network: MAUV Project Update*

Steven G. Chappell
Autonomous Undersea Systems Institute
86 Old Concord Turnpike
Lee, NH 03824
(603) 868-3221
chappell@ausi.org

Roy M. Turner
Elise H. Turner
Department of Computer Science
5752 Neville Hall
University of Maine
Orono, ME 04469
(207) 581-3909
{rmt,eht}@umcs.maine.edu

Charles Grunden
Department of Computer Science
5752 Neville Hall
University of Maine
Orono, ME 04469
charles_grunden@voyager.umeres.maine.edu

Abstract

The cooperative behavior of the participants in Autonomous Ocean Sampling Networks (AOSNs) will be critical to the systems' success. The long term open nature of such systems dictates that they must be able to self-organize and reorganize in the face of change. A two-level approach to coordinating multi-agent organization is discussed in this paper. In particular, we present a collection of cooperative behavior protocols which address many of the issues associated with AOSN self-organization. These protocols have been implemented and are being evaluated in a rule-based simulator. This simulator was designed to experiment with the high level aggregate behavior of agents operating in an AOSN. In this year's work we have exploited the simulator's rule-based nature to increase its fidelity by replacing broad scope rules with sets of finer granularity rules. This provides the means for experimenting with how agents contribute to the AOSN's aggregate behavior.

1 Introduction

The AOSN [Curtin et al., 1993] concept is an important new tool for scientists to use in their efforts to better measure ocean processes and, thereby, improve their understanding of how the oceans influence the planetary ecosystem. Successful deployment of such systems will rely on coordinated, flexible, and adaptive behavior among the system's various participants. We are currently in the second year of work on a Multiple Autonomous Unmanned Vehicle (MAUV) project [Turner et al., 1996] directed at investigating such coordinated behavior. This project has as its goal the development of protocols and mechanisms for intelligently controlling AOSNs. We take a cooperative distributed problem solving (CDPS) approach that uses two kinds of organizations: a *meta-level organization* (MLO) that is responsible for self-organization of the AOSN, and a *task-level organization* (TLO) that actually carries out the mission. Our approach is being developed and evaluated using a rule-based simulator that can model the aggregate properties of an AOSN under the control of various protocols. This approach has several benefits, including: the rule-based methodology provides for rapid development; simulating aggregate properties of the protocols rather than individual participants and their actions allows us to focus on the interaction protocols rather than lower-level control issues; and such a simulator allows us to replace low-fidelity rules with more specific rules as our understanding of the problem and protocols develops.

This paper describes the current state of the project, focusing in particular on the state of the protocols as they are implemented in the simulator. In the next section, we discuss the assumptions about AOSNs un-

*This work was funded by Office of Naval Research contract number N0001-14-96-1-5009 and National Science Foundation grant number NSF BES-9222146.

derlying our work. Following that, we discuss the two-level approach to AOSN organization and operation, and then introduce our protocols for controlling it. After that, we provide an overview of the simulator, with subsections on protocol implementation and environmental simulation. The paper ends with conclusions and future directions.

2 Assumptions

We make several assumptions concerning the domain of AOSN control. The following paragraphs introduce these assumptions as well as the AOSN control requirements they generate.

Autonomous operation. The ultimate missions envisioned for AOSN technology are those characterized by a long-term presence in remote locations coupled with adaptive sampling in response to environmental events. This generates the requirement that the system be able to maintain an effective organization of its participants in the face of change. It should not require constant human intervention, although periodic guidance will be beneficial.

Uncertain initial configuration. Since the system will have to reorganize itself from time to time over any mission, it makes sense to include the ability of *initial self-organization* in the list of AOSN requirements. In fact, the initial self-organization is really a special case of reorganization: creating an organization from a situation where none exists. This ability will have important side benefits. As AOSN participants gather to begin an exercise, the overall system will be able to adjust to deployment accidents, e.g., where air dropped components are damaged or those that were supposed to transit to the work site do not arrive. This means that the AOSN must be able to discover its composition and capabilities at run-time.

Heterogeneous agents. The AOSN participants will cover a wide spectrum of physical and "cognitive" capabilities. We use the term "agent" to refer to any vehicle or instrument platform (VIP) that is capable of making decisions and/or taking independent action. Thus, agents can range from sessile communications nodes to free swimming platforms of various capabilities. In particular, we differentiate between those agents with the intelligence to actively plan, build, and direct the organizing activity, and those agents who generally follow the roles assigned to them.

Long-duration missions. AOSNs will be fielded in order to gather data concerning long-duration ocean processes. This has implications for the composition of the system over time as well as for the need to track and respond to environmental changes over time. In particular, agent organization will have to evolve in response to these changes.

The AOSN is an open system. It is open in that the composition of the set of AOSN agents will not be constant over time. Given the long-term missions envisioned, agents will have to periodically drop out of active service in order to replenish or conserve their energy stores. Agents will also suffer failures or perhaps be required elsewhere and, thus, exit the system. Agents join the system when they finish an energy replenishment cycle. Additionally, agents may be freshly deployed into (and retired from) an operational AOSN as scientists see fit. This open nature has profound implications for AOSN organization and operation, since the AOSN will need to reconfigure itself to respond to its changed organization.

Changing environment and mission. Not only will the AOSN have to operate with a changing agent pool, its chief mission will likely involve adaptive sampling, where the specifics of actions taken by each agent will vary according to environmental change and cannot be fully specified ahead of time. The long term nature of AOSN deployment raises the probability that scientists will redefine mission objectives sometime during that mission. These sorts of changes will impact the AOSN in that it may need to reorganize its agent control structure in order to meet the new situations.

Planning: commitment to future actions. Agent rendezvous, specific spatial and temporal requirements on data collection, known agent entry/exit times, etc. all dictate that the AOSN be able to plan for future actions and events. This argues against the selection of control mechanisms that rely solely on local control of agents, hoping that globally-coherent plans and behaviors will emerge.

3 Organizing AOSN Agents

We take a two-level approach to the problem of controlling AOSN organizations. One level is the TLO, an organization structure composed of some or all of the VIPs, which coordinates and controls the accomplishment of mission tasks. This organization is highly tailored to the AOSN's current situation and the mission at hand. At a higher level is the MLO (cf. [Durfee

and Lesser, 1987]), which is composed of a subset of the VIPs—in particular, those with the capability to participate in a CDPS system. The MLO's purpose is to design an appropriate TLO for the current AOSN, its situation, and its mission. It is also tasked to recover from errors and TLO–task mismatches, which are too severe for the TLO to accommodate.

This two-level approach, we believe, allows sidestepping the usual generality–efficiency trade-off involved in most problem solving. These levels operate at either ends of that trade-off. The MLO is very general and makes minimal assumptions about the AOSN (e.g., which agents are present) and the mission. On the other hand, the TLO can be efficient and inflexible, since any errors or mismatches beyond its ability to handle can be dealt with by the more general MLO.

The two organizations are created and maintained by the AOSN's VIPs following rule-like or script-like patterns of behavior called *protocols*. We are currently experimenting with the following protocols:

- Meta-Level Organization Formation
- Meta-Level Organization Discovery of Resources
- Task-Level Organization Formation
- Task-Level Organization Work Phase
- Entering an Organization
- Exiting an Organization
- Reorganization

The first four flow more or less from one to another in a sequence from an agent's deployment to its integration into the AOSN. The last three protocols handle "exceptions" to that sequence, which are a consequence of the open quality of an AOSN.

In the following subsections, we discuss the current set of protocols in some detail. In the next section, we describe our rule-based simulator in which we are developing and evaluating these protocols. More detailed descriptions of these protocols may be found in [Turner and Turner, 1997].

3.1 Meta-Level Organization Formation

After initial deployment, AOSN agents will have to self-organize into an effective problem solving system based on the agents present, the mission, and the environment. We argue that self-organizing agents will be better equipped to handle the rigors of autonomous or semi-autonomous operation (a primary AOSN requirement) than those that must be provided *a priori* with organizational knowledge. We call the protocol for this activity *MLO formation*. This protocol is carried out by MLO capable-agents—that is, by agents capable of planning, building, and directing the MLO. These are the ones that have the requisite capabilities for cooperative distributed problem solving. MLO formation involves two main steps. First, these MLO-agents must determine whether or not an organization (of any kind) already exists. Second, if no MLO is found, then one of the MLO-agents must initiate the creation of same. This protocol overlaps significantly with the protocol for AOSN reorganization, as discussed in a later subsection. The details of the MLO formation protocol are

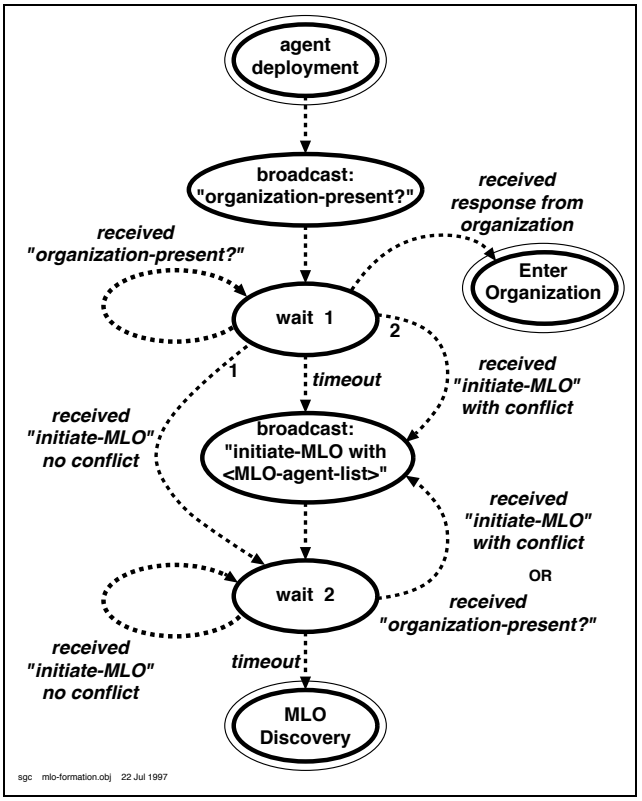

Figure 1: Protocol for MLO Formation.

shown in Figure 1.

MLO formation starts when freshly deployed agents interrogate their environment by broadcasting the message `organization-present?`. This serves both to identify each new agent and to probe for the existence of some AOSN organization. After the broadcasts, each agent sets a timer and enters a wait state. During this wait, an MLO-agent will respond to three kinds of messages. Receiving `organization-present?` messages indicates to it that there are other MLO-agents out there in a state matching its own. It uses data

in the received messages (sender name, location, and MLO capability) to build up its world view regarding the presence and identity of those other MLO-agents.

Receiving an `initiate-MLO` message in this first wait state indicates that another agent is further along in the MLO formation protocol. The receiver must adjust to this situation by moving its state forward within the protocol. How it does this is determined by the match between the receiving agent's world view and the list of MLO-agents it acquires from the `initiate-MLO` message (see arcs labeled "1" and "2" in Figure 1). This MLO-agents list specifies which agents are intended to be the participants in a proposed MLO. If the agents that the recipient already knows about are a subset of those found in the proposed MLO-agent list, then the recipient updates its own knowledge to include those MLO-agents it has just learned about from the list. While this is a conflict, it is one that can be corrected locally, and, thus, it is treated as a non-conflict: the recipient moves directly to wait state 2 (arc "1"). On the other hand, if the recipient knows about agents that are absent from the proposed MLO-agent list, then a conflict exists that cannot be corrected locally. The recipient moves along arc "2" to broadcast its own `initiate-MLO` message with an updated MLO-agent-list in order to correct the sender's knowledge. Otherwise, the recipient's knowledge matches the proposed MLO-agent list; there is no conflict, and it moves to wait state 2 (arc "1"). Finally, receiving a response from a pre-existing MLO or TLO in wait state 1 tells the receiver that it should abandon the MLO formation protocol altogether and enter another protocol in order to join that organization (discussed below).

An agent simply timing out of wait state 1 assumes that it is the first to do so and it broadcasts the `initiate-MLO` message. This announces its unilateral decision to build an MLO made up of the agents it came to know about via the previous reception of the `organization-present?` messages. After this second broadcast, it enters a second wait state, during which, it listens for responses of three types. Reception of `initiate-MLO` messages, which contain information that conflicts with its internal knowledge (i.e., the received message does not contain all the VIPs that this agent knows about), will cause it to attempt to correct the problem by repeating the last broadcast (with an updated agent-list, if necessary) and wait. Reception of `organization-present?` messages will cause the same cycling since they also indicate the presence of previously unknown MLO-agents. Reception of `initiate-MLO` messages containing information that does not conflict with its internal knowledge allows the receiver to remain in the second wait-state until timeout (possibly updating its local information as such messages are received). Through this mechanism, all MLO-agents arrive at world views which are in agreement. They then pass through their wait 2 timeouts, decide that the MLO has been formed, and enter the next protocol.

3.2 Meta-Level Organization Discovery of Resources

Once the MLO has been formed, the system enters the *MLO discovery* protocol, which is characterized by the various agents (MLO and non-MLO alike) arriving at an understanding of where the non-MLO-agents are located, what all agents can do, and who controls whom. The flow through this protocol is shown in Figure 2.

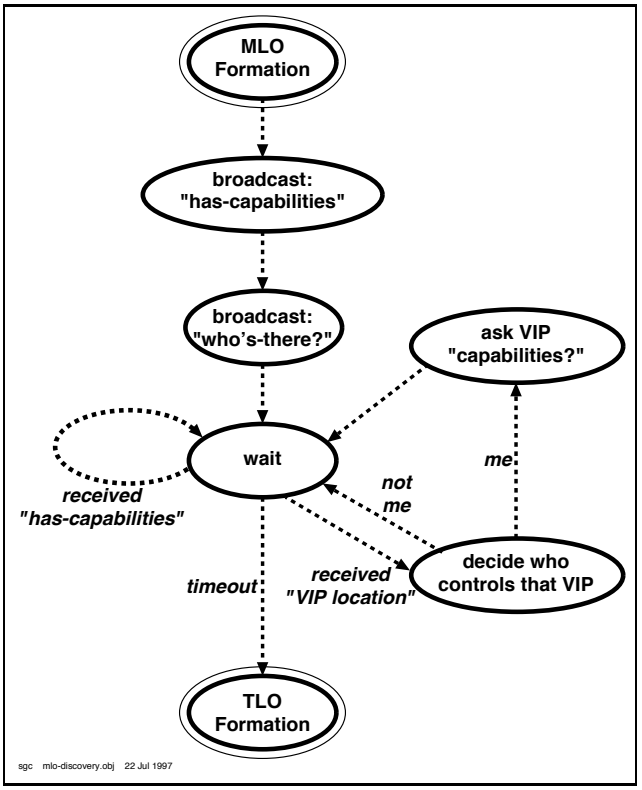

Figure 2: Protocol for MLO Discovery.

MLO discovery is started by each MLO-agent broadcasting a `has-capabilities` message listing what it can do. They also broadcast a `who's-there?` message, which will induce non-MLO-agents into making their capabilities known. After the two broadcasts, the agent enters a wait state to collect replies. During the wait, the broadcaster can respond to either of two inputs. When it receives `has-capabilities` information

from other agents, it updates its own knowledge base so that it can know which agents have what capabilities. Reception of `VIP-location` messages causes it to decide whether or not it should assume control over that responding VIP (which is a non-MLO-agent).[1] Since MLO-agents already have common knowledge of locations, this can be done without communication. If the MLO-agent decides it does control the responding VIP, it asks that VIP about its capabilities, and returns to the wait state. Otherwise it ignores the VIP and reenters the wait. This "capabilities of others" knowledge is used later, when a planner needs to know what capabilities the various MLO-agents have under their control. When the wait times out, it is assumed that all VIPs in the system that can answer have done so. This signals the completion of the formation of the MLO.

3.3 Task-Level Organization Formation

With an MLO completely formed, the system can turn its efforts to the details of TLO construction, in which agents are assigned subtasks of the AOSN mission. This is controlled by the *TLO formation* protocol, which is shown in Figure 3. Note: unlike previous Figures, this one shows *two* chains of state changing. The dashed lines depict the transmission of a message from an agent progressing through states in one chain to an agent progressing through states in the other.

Our TLO-formation protocol is based on a multi-agent planning model (e.g., [Georgeff, 1984; Cammarata et al., 1983]) where a planner is chosen by a simple convention.[2] This will evolve towards the ability of the entire MLO functioning as a planner. Once the planner is chosen, it determines the capabilities required for the mission and then broadcasts a `controlled-capabilities?` message. This is a request for other agents to declare the capabilities they know about. While waiting, the planner responds to any received `controlled-capabilities` messages by building up its own knowledge base of agents' capabilities and the capabilities of agents that they, in turn, control.

After a timeout, the next step is deciding which agents will be working on which mission components. Allocation of resources (agents) to the individual subtasks making up an AOSN mission is complex enough to warrant its own study. As a place-holder for the result

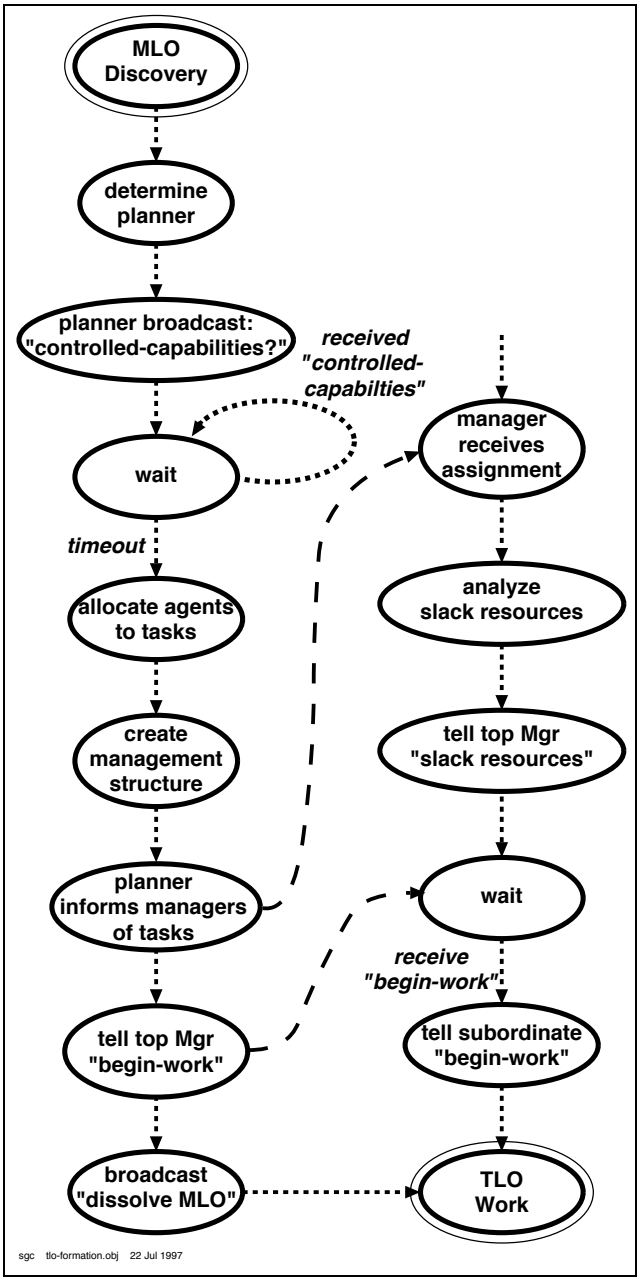

Figure 3: Protocol for TLO Formation.

of that study, we have implemented a simple first-fit algorithm to assign VIPs to tasks. A newer algorithm based on Constrained Heuristic Search (CHS) is being developed and is reported in [Turner, 1997].

After allocating agents to tasks, the planner builds a management structure using a set of heuristics, for example: "if M is working on task T and T has no manager, then make M manager of T". Once this is done, the TLO's organizational structure composed of managers and a top manager has been designed. In the future, we will explore other ways of choosing an

[1] We currently use the simple heuristic of the closest MLO-agent acquiring control.

[2] Our current convention is "first known MLO-agent", which simulates arbitrary planner selection by name, or ability, etc.

appropriate organizational structure for the situation based on the situation's features.

The planner then informs the managers of their assignments, the top-level manager is told to `begin-work`, and then the planner broadcasts a `dissolve-MLO` message to indicate that TLO formation has been completed and that it is time to begin the TLO work phase.

Upon receiving their task assignment messages, each manager (right column in Figure 3) in the TLO hierarchy determines what slack resources it has at its disposal, then notifies the top manager of same, and then waits for the `begin-work` message from the planner. When that is received, managers tell their subordinates to `begin-work`, and they begin their own work. This ends the TLO formation protocol.

3.4 Task-Level Organization Work Phase

The *TLO work protocol* has as yet received little attention, as we are mostly concerned with the dynamics of MLO and TLO organization/reorganization. Some work has been done as a student class project to develop rules to simulate simple tasks being performed by agents in the TLO. The tasks simulated were movement and data sampling tasks in which the simulated sensor was a CTD (conductivity, temperature, and depth) instrument. Another student will work on simulating this phase of AOSN operation in the near future. Again, the goal of this will not be high-fidelity simulation of agent behavior, but rather simulation to such a level that will allow us to gather data about the impact on mission goals of different organization/reorganization protocols.

3.5 Entering an Organization

Agents newly deployed within an operating AOSN will have to follow certain steps in order to join it properly. Likewise, agents that were once members of an AOSN and have been for various reasons "disconnected" from it (replenishing energy, honoring built in down-time constraints, etc.), will have to follow similar steps in order to rejoin. How the agent joins is dependent on the state of the AOSN, as shown in Figure 4.

When deployed (or re-deployed), an agent announces its readiness to join by broadcasting the `organization-present?` message. We have already seen the inner workings of MLO Formation in Figure 1; they have been reduced to a single arc (labeled as number 1) on the extreme left of Figure 4.

Arc number 2 shows what happens when an agent attempts to join an MLO that is already in its discovery phase. If the joiner is an MLO-agent, then the MLO-agent closest to it notifies it that an MLO already exists. Next, the joiner is asked about its capabilities by that MLO-agent. Its `has-capabilities` response allows other agents to discover it as shown by the wait state in Figure 2. Newly joining non-MLO-agents are simply requested to declare their capabilities.

When joining a forming TLO (arc number 3), the nearest MLO-agent to the new agent asks it for its capabilities. The new VIP will respond with its capabilities to the MLO-agent, which will make that information available to the planner when asked for the capabilities it controls. If it has already been asked, then it will immediately inform the planner of the existence of the new capabilities. In either case, the planner uses this new information about the joiner to possibly modify its plans.

When joining a working TLO (arc number 4), the new agent must be fit into the existing management structure. This is accomplished via short dialogs with a manager and the top manager. The state changing traces of these two other agents are shown in Figure 4 to the top and right of arc number 4. Message travel between the traces is shown by dashed lines. Upon receiving the new agent's `organization-present?` message, the nearest manager to it will notify the joiner that a TLO already exists (see top trace). The manager will then notify the top manager about the new agent. The top manager will then request the new agent to report its capabilities (see extreme right hand trace). When that information becomes known to the top manager, it can decide when to tell the new agent to `begin-work`.

3.6 Exiting an Organization

When an agent must leave an AOSN, it should do so in a graceful manner so that its absence does not become detrimental to the AOSN, at least not catastrophically so. An *exiting protocol* is shown in Figure 5.

Obviously, not all exits will be graceful. Agents can suffer various levels of failure which would prevent them from participating in the graceful exit protocol. In those cases, the remaining AOSN must first detect the agent's exit and then adjust to it. Detection might be effected by noticing the missing agent's non-responses to communication attempts, for example. Once this absence is known, managers and planners can exam-

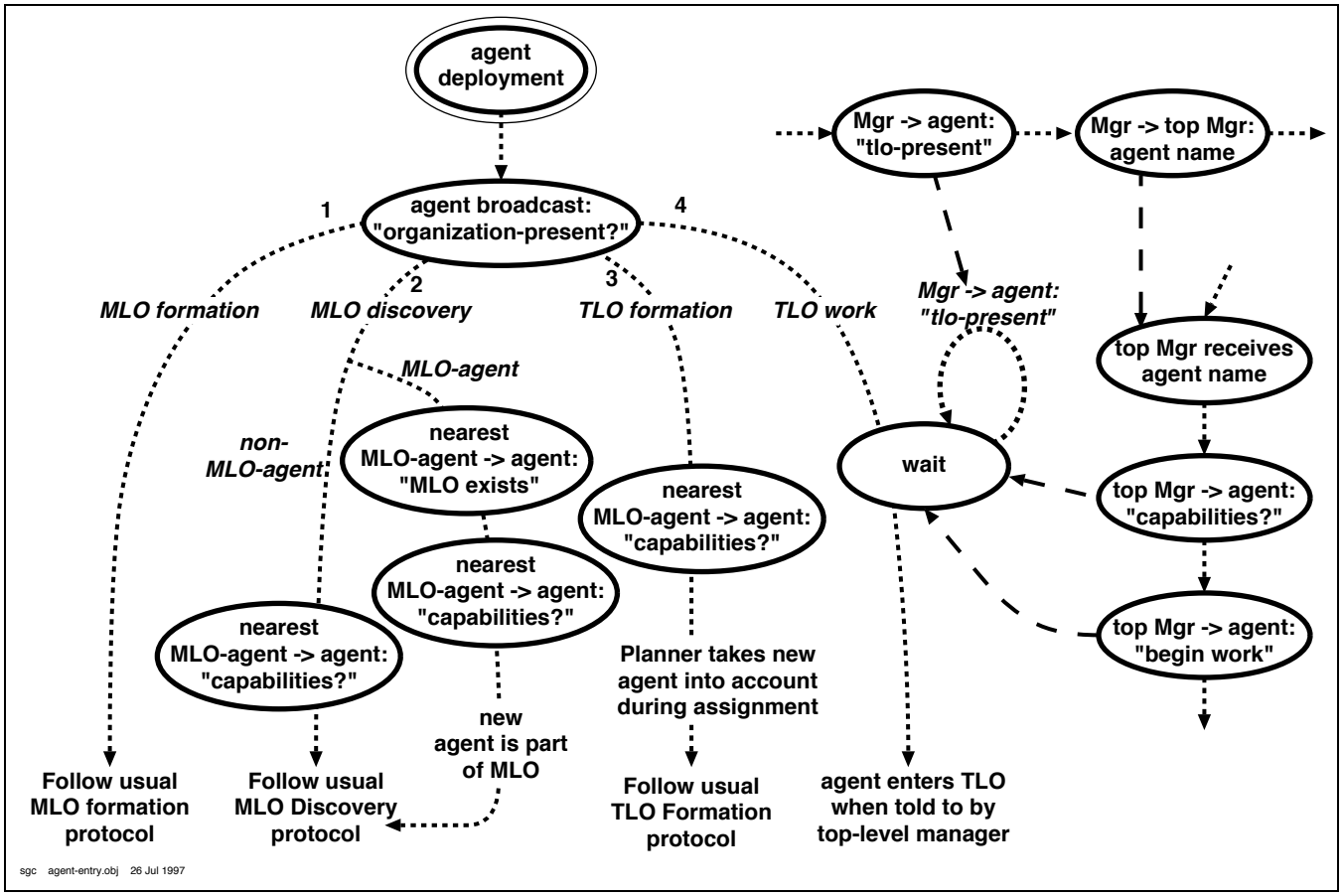

Figure 4: Protocol for Entering an Organization.

ine their slack resources knowledge and develop replacement strategies. Of course, failure recognition can be more subtle than this, making this area a target for future work.

Once the agent failure is known, the top level manager will try to adapt the TLO to the new situation. Failures of non-managers should result in "simple" replacement from the slack resources pool. Replacement of managers is more involved in that the missing management functionality must be replaced as well as any capabilities needed for the mission work itself.

3.7 Reorganization

In cases where managers cannot adapt the TLO to a developing situation, it is time start the *reorganization protocol*. Failure to adapt can be caused by many things: failure of a top level manager, lack of a replacement agent with the proper capabilities for a "must do" mission task, arrival of new mission tasks, changes in the environment, changes in situation that provide opportunities for a more efficient TLO, completion of one or more mission tasks, and AOSN user intervention.

A simplified version of the reorganization protocol is shown in Figure 6. When an MLO-agent believes that reorganization is necessary, it enters into negotiation with its peers. If the peers agree, then it initiates the re-formation of the MLO to handle the reorganization. If one or more peers do disagree, however, then (at the present) it aborts the reorganization. This is not ideal, and we are still working on protocols to handle peer disagreement.

4 The Simulator

Our simulator was designed to focus on the top level aspects of organizing, managing, and performing AOSN missions. We purposely avoided the problems associated with simulating the low level activities of the individual AOSN agents. This has allowed us to simulate the AOSN at a very high level, though at low fidelity,

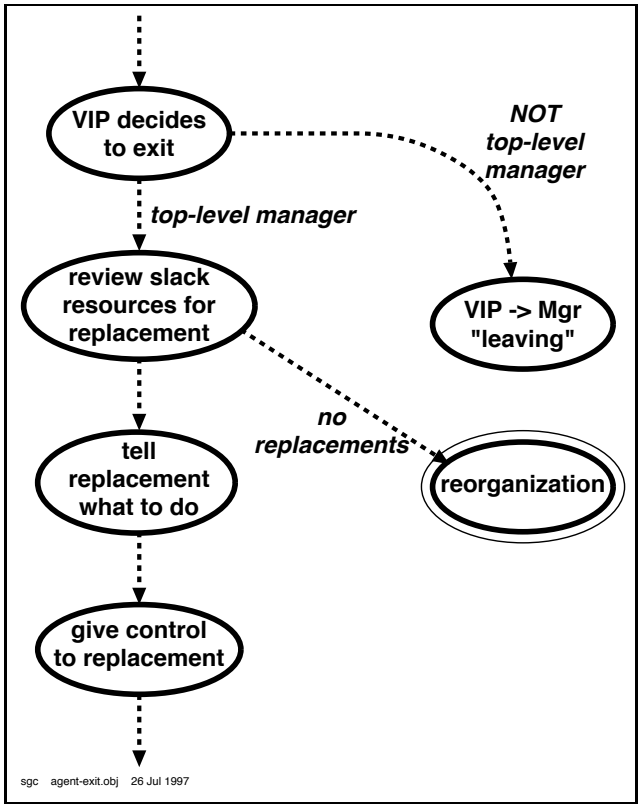

Figure 5: Protocol for Exiting an Organization.

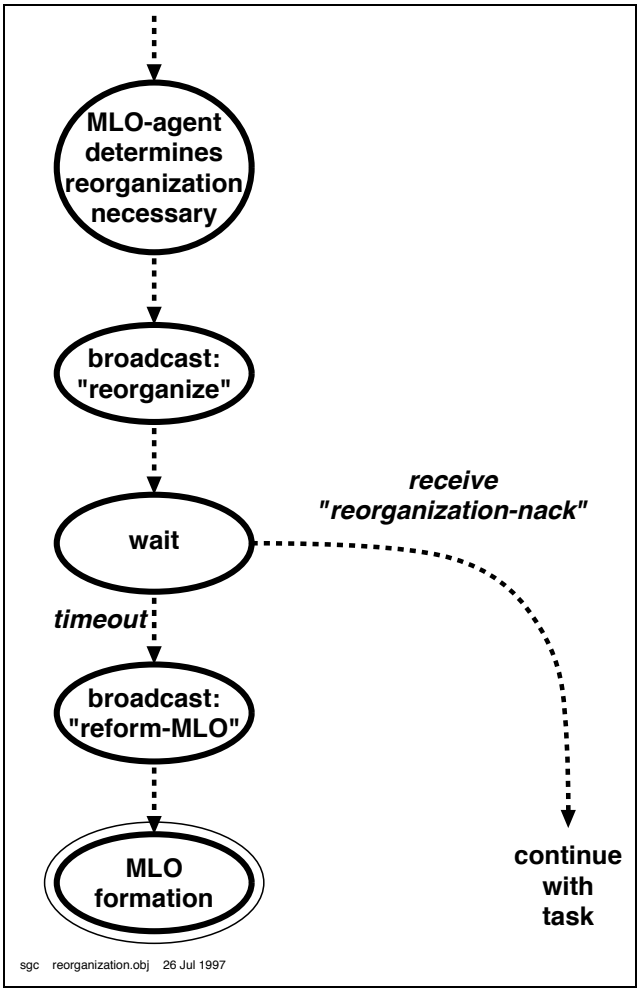

Figure 6: Protocol for Re-Organizing.

to quickly get an idea of how the protocols are working. Since most of the simulator's code is expressly for protocol implementation, when something fails it is the protocol failing and not some underlying agent simulation, on which, the protocol might have been relying. This allows us to focus on the protocols themselves.

The simulator has been implemented using NASA's C Language Integrated Production System (CLIPS) expert system language shell and environment [Giarratano, 1993]. CLIPS' portability allows us to run the simulator on both SPARC/Solaris and Intel/Linux platforms. The current simulator rule base contains approximately 260 rules. In addition to supporting rule sets, the CLIPS environment also allows for the inclusion of custom functions written in the C language. These custom functions become part of the CLIPS inference engine and are then callable by rule antecedents and consequents. This feature provides a means for installing algorithms that are not suitable to rule-based expression. We have used this feature to implement the previously mentioned simple first-fit algorithm for agent to AOSN subtask allocation.

4.1 Simulating the Protocols

Protocol simulation is relatively straightforward in the simulator, since our protocols have a rule-like flavor. Each of the major protocols described above are simulated using a set of rules comprising a "phase" of the simulator's operation by using one antecedent of each rule to specify the phase. This is a typical context-limiting mechanism in rule-based systems.

The original scope of the simulator was in the realm of the abstract, high-level aspects of aggregate group behavior, not the concrete, low-level details of individual agent behavior. Thus, the system started out being a low-fidelity simulator dealing with organizing, reorganizing, and managing AOSN missions. To achieve the high-level simulation desired, we try to avoid rules having to do with the internal decision-making processes of the VIPs. Instead, we concentrate on simulating the *effects* of the decision making. We can do this because the simulator itself has a global perspective and because

we know what the protocols specify the VIPs' behavior should be.

During this year's development, we have been exploiting the flexibility offered by the rule-based paradigm to move the simulator toward a higher degree of fidelity. This was done by replacing abstract rules having broad scope with more concrete rules having more specific effects. This moves us closer to the next phase of simulation, in which the VIPs' control algorithms will be directly implemented in simulation.

4.2 Simulating the Environment and Vehicle Motion

In addition to simulating the protocols, the simulator also needs to simulate the AOSN's environment and the movement of its VIPs as they go about carrying out the mission. This is done by adding environment and motion rules to the simulator, as well as rules to simulate the VIPs' sensors. These rules make use of the discrete-event simulation capabilities of our simulator, as discussed below. In the future, we will explore connecting the simulator to higher-fidelity environmental models written in other languages (e.g., C++) via network connections.

The movement rules fire each time the simulator's time changes due to any event occurring. These rules have a high priority, so that the vehicles' positions are updated before simulating any other actions or sensor events that might be scheduled to occur at the current time. There are three major types of rules for movement: velocity, turning, and collision detection rules. The velocity rules determine the next position of each vehicle by first calculating the effect of its velocity vector on its position, then calculating the effects of the yaw, pitch, and roll on that vector. Finally, the vector is used together with the current position of the vehicle to yield the next location. Due to the discrete nature of the simulated time, smooth turning is not simulated. Instead, the turning rules work by first changing the vehicle's heading halfway to the desired heading. Then the velocity rules fire, moving the vehicle. Lastly, the turning rules complete the vehicle turn. This is not an accurate representation of the turn, but it is a close enough approximation at this time. The collision detection rules check the path of each moving vehicle for possible intersection with another vehicle's path.

Unlike the movement rules, which fire every time the simulator's time changes, sensor rules fire based on the occurrence of events. This simulates the actual sensors by only updating when the actual sensor normally would update. When a sensor is turned on, an event is posted for some future time when the sensor data would be obtained, based on the sensor's cycle rate. When the event occurs, a sensor rule fires, which determines what the VIP's sensor would "see" at the current time and location and puts that fact in the simulator's working memory. Last, it posts another event to occur after the amount of time has passed that it normally takes that sensor to actually update. The sensors that are currently implemented are: water temperature, depth based on pressure, depth and altitude based on an up-down sonar, magnetic compass, position from long-baseline navigation, and velocity.

As an example, suppose a vehicle named EAVE-Arista is traveling at a speed of 1.5 meters per second. It is also changing its heading at a rate of 10 degrees per second and its altitude sensor is turned on. Suppose that the next scheduled event is for EAVE-Arista's altitude sensor to update in one second. Since there are no more events, the simulator selects this one and increments the time by one second. Before the sensor rules that update the sensor fire, the movement rules trigger due to the change of system time. The turning rules change EAVE-Arista's heading halfway; by 5 degrees. Next, the velocity rules move the vehicle forward by 1.5 meters, and the turning rules change the heading again by 5 degrees. The collision detection rules then fire, checking to see if it hit the bottom or any other agent. Finally, the altitude sensor is updated according to EAVE-Arista's new location.

5 Conclusions and Future Work

Intelligent control of an AOSN is a difficult task. It requires that the AOSN be capable of organizing itself and reorganizing as its environment, mission, or composition changes.

In the MAUV project, we are developing protocols to implement a CDPS approach to this problem. In this phase of the project, we have developed protocols for: formation of a meta-level organization, discovery of resources, design and creation of an appropriate task-level organization, and reorganization of the system when needed. We are developing and evaluating our approach in the context of a rule-based, aggregate level simulator. This allows us to simulate the gross properties of an AOSN following our protocols without getting bogged down in the details of how each VIP in the system decides what to do next. This simulator has

proven to be useful for rapid development of protocols and for quickly evaluating different ideas.

In the near future, we will continue developing the current set of protocols and continue their evaluation via empirical studies in the simulator. In the next year, we will begin to move toward the next major phase of this simulator's development. This involves simulating the AOSN at the level of individual agent decision making. In addition to augmenting the simulator's abilities with new rules (where appropriate), we plan to add network interfacing functions as well. This will allow it to communicate with other simulation facilities under development at the University of Maine and the Autonomous Undersea Systems Institute. At the moment, we are experimenting with using the Knowledge Query and Manipulation Language (KQML) [Finin et al., 1995] as the language for communication between the various simulators. We will also continue development of the TLO work phase to support the simulation experiments.

References

Cammarata, S., McArthur, D., and Steeb, R. (1983). Strategies of cooperation in distributed problem solving. In *Proceedings of the 1983 International Joint Conference on Artificail Intelligence*, pages 767–770.

Curtin, T. B., Bellingham, J. G., Catipovic, J., and Webb, D. (1993). Autonomous oceanographic sampling networks. *Oceanography*, 6(3):86–94.

Durfee, E. H. and Lesser, V. R. (1987). Using partial global plans to coordinate distributed problem solving. In *Proceedings of the 1987 International Joint Conference on Artificail Intelligence*, pages 875–883.

Finin, T., Labrou, Y., and Mayfield, J. (1995). KQML as an agent communication language. Available at URL http://www.cs.umbc.edu/lait/papers/kqml-acl.ps (1 May 1997). Also: "to appear (1995) in Jeff Bradshaw (Ed.), *Software Agents*, MIT Press, Cambridge".

Georgeff, M. P. (1984). A theory of action for multi-agent planning. In *Proceedings AAAI-1984*, pages 121–125.

Giarratano, J. C. (1993). *CLIPS User's Guide*. NASA, Information Systems Directorate, Software Tehcnology Branch, Lyndon B. Johnson Space Center, Houston, TX.

Turner, E. H. (1997). Task assignment in AOSNs: A constraint-based approach. In *Tenth International Symposium on Unmanned Untethered Submersible Technology*, September 1997, Durham, NH. Autonomous Undersea Systems Institute, 86 Concord Turnpike, Lee, NH 03824.

Turner, R. M. and Turner, E. H. (Submitted April 1997). Adaptive organization and reorganization of autonomous oceanographic sampling networks. *Journal of Applied Intelligence*.

Turner, R. M., Turner, E. H., and Blidberg, D. R. (1996). Organization and reorganization of autonomous oceanographic sampling networks. In *Proceedings of the 1996 Symposium on Autonomous Underwater Vehicle Technology*, pages 407–413, June 1996, Monterey, CA. IEEE Ocean Engineering Society, IEEE Publishing Services.