A Two-Level, Protocol-Based Approach to Controlling Autonomous Oceanographic Sampling Networks

Roy M. Turner and Elise H. Turner
Department of Computer Science
5752 Neville Hall
University of Maine
Orono, ME 04469
Phone: 207-581-3909 E-mail: rmt@umcs.maine.edu

Abstract

Autonomous oceanographic sampling networks (AOSNs) are groups of autonomous underwater vehicles and other instrument platforms. They were envisioned to address the serious undersampling problem that exists for the ocean. Advanced AOSNs will include numerous heterogeneous components and perform complex, long-duration missions. They will have to cope with vehicles failing, new vehicles arriving, and changes in the mission and environment. Controlling such a system is a very challenging task.

The CoDA project focuses on developing intelligent control mechanisms for advanced AOSNs. CoDA treats AOSN components as agents that follow protocols to create an effective organization to carry out the mission. CoDA is a two-level approach. A *meta-level organization* (MLO) first self-organizes from a subset of the agents to discover the AOSN's total capability repertoire, then designs a *task-level organization* (TLO) to efficiently carry out the mission. When changes occur, the MLO can step in to reorganize the AOSN to fit the changed situation. This two-level approach allows the AOSN to be both efficient and flexible in the face of change.

In addition to describing CoDA, this paper also presents simulation results to show its behavior, both normally and when agents fail.

Index terms—Autonomous oceanographic sampling networks, autonomous underwater vehicles, multiagent systems.

1 Introduction

Autonomous oceanographic sampling networks (AOSNs) [1] are multi-component systems being developed to collect data for ocean scientists. They were envisioned to address the serious undersampling problem that exists for the ocean. Groups of vehicles and instrument platforms (VIPs) can sample an area much more efficiently than a single vehicle, potentially remain on-site longer than a human-crewed vessel, and take simultaneous samples at different locations to provide a synoptic view of an area. AOSNs and AOSN-related technology have applications in other areas as well, such as surveillance and mine countermeasures.

Creating an AOSN is a formidable challenge that pushes the boundaries of the state of the art in vehicle systems and hardware, especially with respect to control software for the multiagent system. The AOSNs that have been developed and fielded so far [e.g., 2; 3] are relatively simple, generally consisting of only a few autonomous underwater vehicles (AUVs) cooperating under close human supervision in order to test key AOSN technologies and techniques, often while providing useful scientific data.

To go much beyond the current generation of AOSNs, new, sophisticated control mechanisms must be developed to allow the components to cooperate to form a capable whole. In this paper, we are concerned with advanced AOSNs that will be able to undertake complex missions and be capable of operating autonomously for an extended period of time. Such an AOSN will be comprised of a large, heterogeneous collection of vehicle and instrument platforms. Although the AOSNs we envision will have enough intelligence, in the aggregate, to function as assistants to scientists and other users, their individual components will have a wide variety of capabilities. These capabilities will include physical capabilities, such as sensing and movement capabilities, as well as the intellectual capabilities needed to organize and manage the AOSN. These advanced AOSNs will be capable of being deployed for months or years. Consequently, such an AOSN will be an *open system* [4] that components may enter or exit at any time.

Our work focuses on developing artificial intelligence techniques needed for such an AOSN. We believe that a key to this is recognizing that the AOSN control task is largely one of selecting and maintaining an appropriate *organization* in which components operate. It is highly desirable that the AOSN be able to accomplish this autonomously, for several reasons. There are many missions in which communication with a user or controller is not possible or is ill-advised, for example, covert mine hunting. Also, it is often impossible to know ahead of time what the conditions will be at the work site, and so designing an organization *a priori* is out of the question. It also makes little sense to burden a scientist or other user with the task of designing the AOSN's organization, since selecting the proper organization requires a great deal of knowledge about the task to be carried out, the environment, and the VIPs comprising the system. Finally, the situation will undoubtedly change over the course of the mission: vehicles will fail, the user will change the mission, the environment will drastically change, or new vehicles will arrive and need to be integrated into the AOSN. When this happens, the AOSN needs the ability to reorganize to address the changes.

The CoDA (Cooperative Distributed AOSN control) Project [5; 6] is developing techniques for organization and reorganization of advanced AOSNs. We have taken a two-level approach to address both flexibility and efficiency, two concerns that are often viewed as incompatible within an organization. A *meta-level organization*, or MLO, first self-organizes from a subset of the VIPs present using a set of shared protocols. It gathers information about the AOSN's capabilities, the environment, and the mission. Its members then collaborate to design a *task-level organization*, or TLO, to fit the current situation.

The MLO is similar in some ways to a board of directors. It is composed of the most intelligent agents, but most likely, there will be relatively few of them in the AOSN. It constructs a TLO, then leaves it to do its work until a new organization is required. The TLO, on the other hand, is composed of all the agents in the AOSN, which includes a much wider range of VIPs. Some may be simple non-mobile sensors which can only return specific kinds of data at a particular location, while others may have data collection capabilities as well as the reasoning capabilities required to manage other agents. The MLO assigns these agents to specific roles in the TLO where their capabilities are needed.

The TLO then carries out the mission until there is a change in the situation. It has a limited ability to adapt to changes, since it must carry out the mission while making efficient use of the AOSN's resources. It may be able to reassign roles, but for significant changes, the MLO must step in to reorganize the AOSN by designing a new TLO to fit the changed situation.

The CoDA Project currently focuses on three aspects of this two-level approach. First is a set of protocols for VIPs in the AOSN to use. These are *cooperation protocols*, much as TCP/IP is a communication protocol: they are essentially rules about how an agent should respond to actions taken by others or to environmental and mission features. So far, work has concentrated on organization/reorganization protocols. Second is a task assignment mechanism that matches agents' capabilities to the needs of roles in an organization. And third is an organization selection/design mechanism that chooses or creates an appropriate organizational structure (e.g., a hierarchy, team, market, etc.) to fit the AOSN's current situation and mission. CoDA is being developed and tested in a multi-fidelity simulator [6] that allows direct simulation of the protocols at a high level of abstraction, while allowing particular techniques (e.g., for task assignment) to be incorporated in much the same form they will take aboard VIPs.

This paper focuses on the protocols. The task assignment mechanism is described in detail elsewhere [7], and work on the organization selection mechanism is still in the early stages. In the remainder of this paper, we first describe a hypothetical AOSN that we will use as an example throughout the paper. In Section 3, we

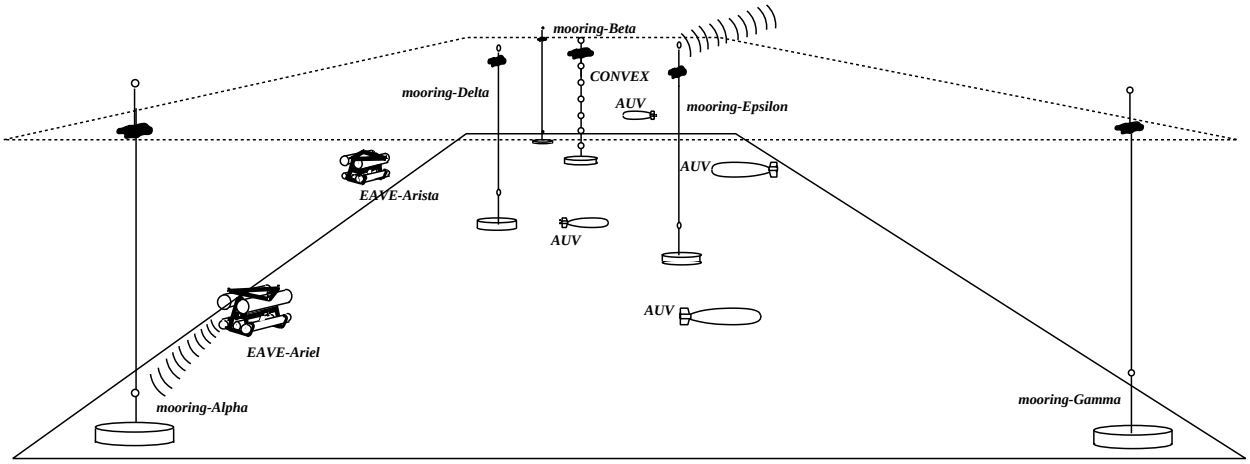

Figure 1: The example AOSN.

describe the CoDA protocols, and Section 4 summarizes our simulation results. Section 5 discusses related work.

2 An Example AOSN

Our hypothetical AOSN is composed of heterogeneous components and has two tasks: generating routine data (e.g., CTD[1] and sonar data) about a region and adaptively sampling a sub-region when a convective overturn event occurs.

Oceanographers and climatologists are interested in convective overturn of the water column in high latitudes [8]. It is hypothesized, for example, that such events play an important role in global climate change. Instruments such as the CONVEX mooring [8] have been developed to detect overturn events. However, a moored string of instruments can at best sample the event in a single dimension over time, making it very difficult to characterize the spatiotemporal structure of the convective plume. In addition, if the mooring happens to be on the edge of the event, it will sample it poorly. An AOSN containing AUVs in addition to a CONVEX mooring offers an ideal solution to this problem. When the mooring detects that a convection event is in progress or about to begin, the AOSN can deploy its AUVs to characterize the event in three dimensions over time. As the plume develops, the AUVs can adaptively sample it by changing their positions, and perhaps the sensors they are using, in response to the event's evolution.

Our example AOSN is shown in Figure 1. It is a heterogeneous system composed of six AUVs and six moorings. We assume that most realistic AOSNs will have VIPs with various capabilities, including various levels of intelligence. Consequently, the example AOSN includes two EAVE-III vehicles [9], named Ariel and Arista, which we assume are equipped with high-level, artificial intelligence-based mission controllers. These AUVs can participate in meta-level protocols for organization/reorganization of the AOSN. The other AUVs do not possess the ability to participate in the meta-level organization, although we assume that they have sufficient intelligence and communication abilities to manage other VIPs within the task-level organization. Ariel has a CTD sensor and can manage up to four other VIPs. Arista has a side-scan sonar and can manage up to five other VIPs. One of the other AUVs has a down-looking sonar, and all of the others are sufficiently intelligent to manage up to four other VIPs.

The moorings consist of a CONVEX mooring, three long-baseline (LBL) navigation moorings (Alpha, Beta, and Gamma), and two radio moorings that break the surface (Delta and Epsilon). None of the moorings can manage, nor can they participate in the meta-level organization.

[1] Conductivity, temperature, and depth.

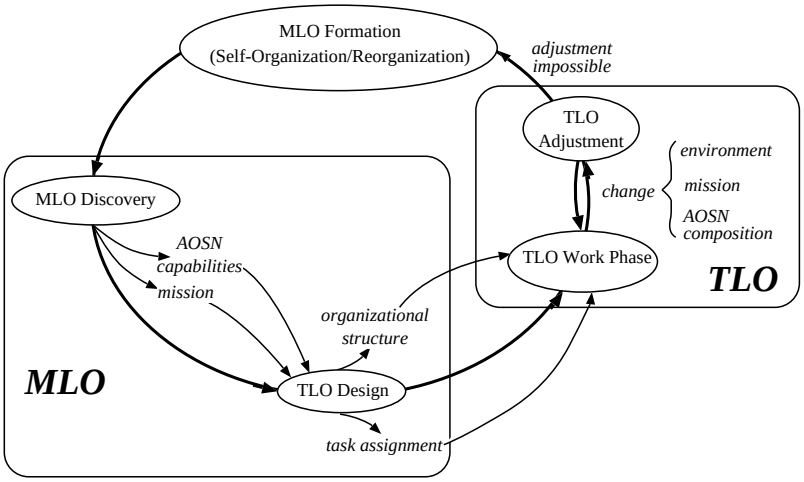

Figure 2: CoDA's overall approach.

3 CoDA Organization/Reorganization Protocols

CoDA's overall approach is shown in Figure 2. When the AOSN is deployed, a meta-level organization is first created to design the organization that will actually carry out the mission. Not all VIPs present may have the necessary intelligence to participate in the MLO. Consequently, the MLO will be comprised of only those that can, called *MLO agents*. The MLO designs the task-level organization, taking into account the mission, environment, and all capabilities of both MLO agents and non-MLO agents. The TLO is then given control to conduct the mission. Minor problems and changes in the situation can be accommodated by the TLO. For example, the TLO might be able to handle the failure of an AUV by reassigning its duties to another AUV, if there are sufficient slack resources in the system. Other times, the TLO will not be able to adapt to the changed situation. For example, if the current plan for conducting the mission relies on a sensor that only one VIP has, and that VIP fails, then the TLO will not be able to adjust to the change. Instead, the MLO will re-form to analyze the mission goals in light of the changed situation. It may be able to successfully reorganize the AOSN, for example, by designing a new TLO to accomplish the mission in a way that does not require the missing sensor.

Agents in the AOSN follow *cooperation protocols* to organize, reorganize, and carry out the mission. A protocol is a pattern of well-defined responses to messages or events, where the responses are themselves messages or actions. The AOSN follows different protocols during different phases of its operation; the MLO, for example, follows one protocol during self-organization and another when attempting to discover the capabilities of the AOSN as a whole.

There are many approaches to cooperation in a multiagent system (see Section 5). We chose protocols for several reasons. Using protocols does not require one agent to have intimate knowledge of another's internal state or structure, as do some other mechanisms. This allows the specifics of VIPs to be abstracted away so that the AOSN control mechanism can concentrate on what the vehicles can do relative to the mission. CoDA protocols treat all VIPs as black boxes with an advertised list of capabilities and resources drawn from a standard vocabulary [see, e.g., 10]. We assume that any VIP that is to participate in the AOSN can, at minimum, communicate its capabilities and resources in this vocabulary in response to a query from another agent. For example, an AUV might respond to such a query with the information that it has the capabilities of side-scan sonar, CTD (conductivity, temperature and depth) sensor, and torpedo-like motion. More abstract capabilities, such as the ability to manage other agents and the ability to follow MLO protocols, are advertised in the same way. This allows the MLO to determine which VIPs should perform which roles in a TLO based solely on their capabilities and resources. Protocols can easily be simulated. The aggregate properties of the AOSN's behavior when agents are following protocols can be modeled without

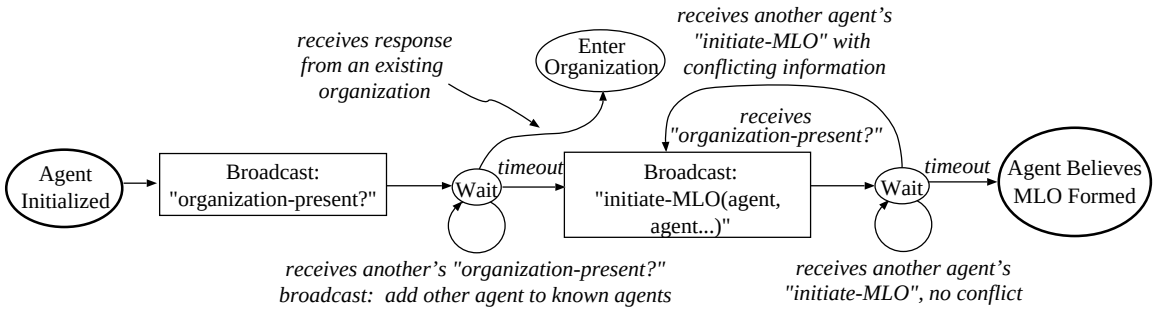

Figure 3: The MLO formation protocol (for MLO agents).

the need to directly simulate the internals of agents. This allows the overall approach to be evaluated early in its development. Using protocols also keeps both the simulation and the approach more general. Different protocols can be designed for different classes of agents, thus helping to satisfy the goal of allowing any VIP, even extremely simple ones, to participate in the AOSN. Details of specific AOSNs, such as properties of their communication network, can be ignored when creating and testing the general approach. The protocols can later be tuned to specific AOSNs or different classes of missions.

In the following, we discuss the current protocols for each phase of an AOSN's operation.

3.1 MLO Formation

After initial deployment, the nascent AOSN's components must first self-organize. This can happen as long as there is one or more MLO agents present. The protocols for this phase causes MLO agents to discover one another and then to form a meta-level organization. For generality, we make few assumptions about how many or what kind of agents are present at deployment. This allows the protocols to handle the cases in which not all agents successfully arrive at the AOSN's work site, or in which the agents arrive over some significant interval of time.

Figure 3 shows the MLO formation protocol followed by each MLO agent. An agent may be unaware initially of the presence of others, and so each one must first try to determine who else is present. This is done by broadcasting an `organization-present?` message, then waiting for responses.[2] As described below, this message is also part of the VIP entry protocol, and if an existing organization (TLO or MLO) replies in response, the agent follows that protocol to join the organization.

The agent then waits for a period of time, listening for other agents' `organization-present?` broadcasts. This is how it determines who its peers are. At the end of its wait period, it assumes that it knows who all the other MLO agents are. It suggests, via an `initiate-MLO` broadcast, that an MLO be formed with these agents. It then waits again to give its peers a chance to agree or disagree.

Disagreement may happen when other agents arrive late, and so broadcast their initial messages after this agent has tried to initiate an MLO. It may also happen if some MLO agents disagree about which MLO agents are really present. This could happen if the communication channel is such that some agents are not able to hear others. It could also happen due to vagaries of timing, allowing, for example, new agents to enter after the request to initiate an MLO has been sent. If the initiating agent receives another agent's `organization-present?` message during this wait period, or if it receives another MLO agent's `initiate-MLO` message that contains information about agents it does not know about, then it updates its own knowledge and sends an updated `initiate-MLO` message. This is done to avoid other agents who have received the agent's first message erroneously believing that the MLO is smaller than it should be. If the initiating agent receives non-conflicting `initiate-MLO` messages, it does nothing. When this waiting period

[2]Currently, few assumptions are made about the underlying communication network other than that it is low-bandwidth and that it supports both broadcasts and point-to-point messages. Error-free communication is assumed, however, as is the case with TCP/IP. This assumption will be relaxed in future work.

```
00:00:00.0 (MLO) new agent EAVE-Ariel broadcasting organization-present?.
00:00:00.0 (MLO) new agent EAVE-Ariel setting timer 1 to wait for replies.
00:00:00.0 (MLO) new agent EAVE-Arista broadcasting organization-present?.
[...]
00:00:01.01 (MLO) EAVE-Arista: received organization-present? message from EAVE-Ariel
00:00:01.01 (MLO) EAVE-Ariel: received organization-present? message from EAVE-Arista
[...]
00:00:30.0 (MLO) EAVE-Ariel: waited long enough for organization-present? replies.
00:00:30.0 (MLO) EAVE-Ariel: initiating MLO formation with agents = (EAVE-Arista
           EAVE-Ariel)
00:00:30.0 (MLO) EAVE-Ariel: broadcasting first initiate-MLO message.
00:00:30.0 (MLO) EAVE-Ariel: setting timer 2 to wait for replies.
00:00:30.0 (MLO) EAVE-Arista: waited long enough for organization-present? replies.
00:00:30.0 (MLO) EAVE-Arista: initiating MLO formation with agents = (EAVE-Ariel
           EAVE-Arista)
00:00:30.0 (MLO) EAVE-Arista: broadcasting first initiate-MLO message.
00:00:30.0 (MLO) EAVE-Arista: setting timer 2 to wait for replies.
00:00:31.01 (MLO) EAVE-Arista: In wait 2: initiate MLO received; no conflict.
00:00:31.01 (MLO) EAVE-Ariel: In wait 2: initiate MLO received; no conflict.
00:01:00.0 (MLO) EAVE-Ariel: completed MLO formation.
00:01:00.0 (SIM) MLO formed, contains members (EAVE-Ariel EAVE-Arista).
00:01:00.0 (SIM) switching context -> MLO discovery
```

Figure 4: Simulator output during MLO formation. Time is shown at the left as hh:mm:ss.s.

is over, the agent believes that the MLO has been formed, with mutual knowledge among the participants about who is part of the organization. At this point, the next phase of operation, MLO discovery, is entered.

In our example AOSN (Section 2), there are two MLO agents, EAVEs Arista and Ariel. These independently follow the MLO formation protocol, resulting in them coming to the shared belief that an MLO should be formed consisting of both of them. Figure 4 shows output from the CoDA simulator during this phase.

3.2 MLO Discovery

One an MLO is formed, it enters the MLO discovery phase to determine the total capabilities of the AOSN. Also in this phase, the MLO determines what the mission is. This is done in case only one or several of the MLO agents were given the information initially.

There are at least two ways to manage the knowledge about the non-MLO agents. The MLO agents can share the knowledge, so that they achieve common knowledge of the AOSN's total capabilities. Alternatively, each MLO agent can be responsible for interacting with and knowing about some subset of the non-MLO agents. We refer to MLOs taking the former approach as "flat" and ones taking the latter as "hierarchical". The current protocol (see Figure **??** is for a hierarchical MLO that uses the location of a non-MLO agent to determine who is responsible for knowing about it. This minimizes cognitive load on any individual MLO agent. Future work will compare the two kinds of MLOs.

A non-MLO VIP replies to the broadcast with a `VIP-location` message containing its identity and location. An MLO agent hearing this message, based on knowledge of its own, its peers', and the non-MLO agent's locations, decides if it should be the one responsible for the VIP or not. If so, then it sends a request for information about its capabilities and stores that information when it is received.

The final part of the protocol is designed to synchronize the MLO agents as they transition to the next phase. Note that an agent can request additional time if it needs it, for example to wait for a reply from a controlled VIP. When all MLO agents agree, then the AOSN will be in the next phase, TLO design.

The protocol for non-MLO agents for this phase is simple. When a VIP receives a request for information from an MLO agent, it just responds with the information.

In the example AOSN, the MLO agents each broadcast messages to determine which other vehicles might be present. In this case, the other AUVs and the moorings all report their identities and locations. Arista and

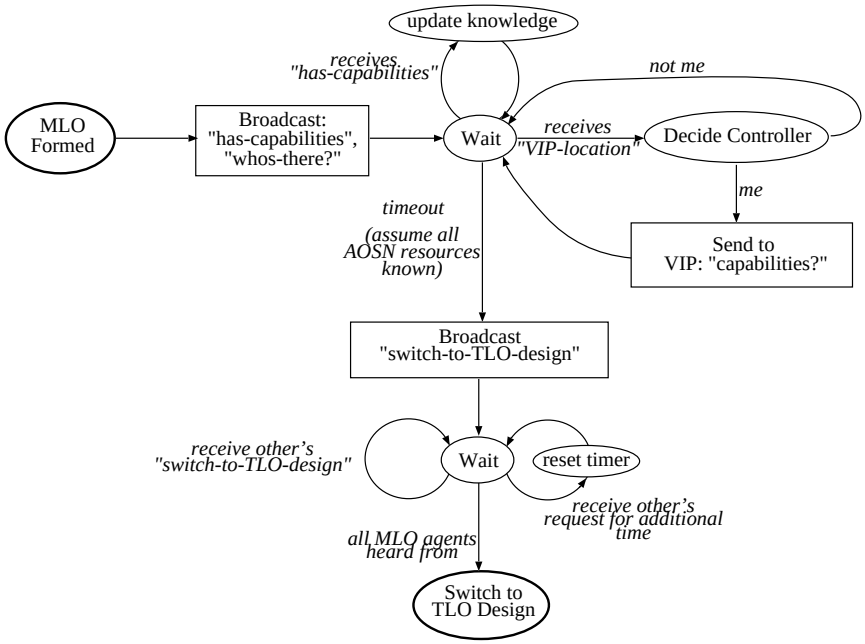

Figure 5: MLO agent protocol for MLO discovery phase.

Ariel then each decide, based on their common knowledge about the locations of other VIPs and themselves, which VIPs they will control. They send messages accordingly, with the result that by the end of this phase, all capabilities of the AOSN are known to either Arista or Ariel. For brevity, we omit simulation output for this phase.

3.3 TLO Design

Once the MLO knows the AOSNs capabilities, the next phase of operation is to design an appropriate task-level organization for the situation. This involves selecting an *organizational structure* and then filling its roles with VIPs having appropriate capabilities.

Ultimately, the TLO design phase will begin with a subset of the MLO agents "brainstorming" about which organizational structure is best for the current situation. An organizational structure is a pattern of roles, authority relationships, and communication types and pathways that defines an organization. Examples include hierarchies, markets, committees, and teams. We are currently developing a mechanism in which an MLO agent uses diagnostic reasoning based on features of the situation to select the organizational structure that is appropriate. This approach draws on the considerable body of literature on human organizational design theory. In the fully-distributed version, MLO agents will have to negotiate to agree on the best organization structure, since they may have different knowledge and perceive the world differently. At the current time, the organizational structure selected by the MLO is always a multi-level hierarchy.

Once an organizational structure is selected, the MLO will assign VIPs to its roles. We are investigating several distributed constraint-based methods for this, including a distributed version of constrained heuristic search [11]. This combines constraint satisfaction techniques with heuristic search techniques into a unified approach. The current version of CoDA uses a single-agent version of this technique, extended for the special requirements of task assignment in AOSNs [7]. In a distributed version, the problem will be attacked by multiple MLO agents simultaneously.

Once role assignments are decided, VIPs are notified of their roles. At this point in the current approach, the MLO dissolves, to be replaced by the new TLO. In the future, we will explore having some form of the MLO remain in existence while the TLO is in control.

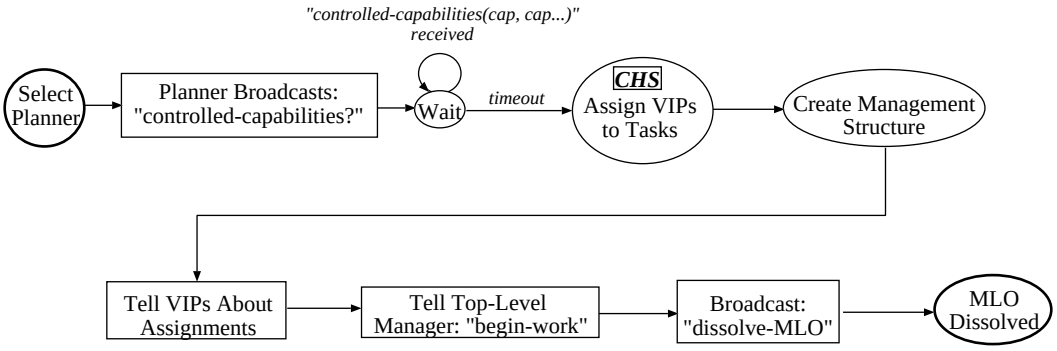

Figure 6: TLO design protocol for MLO agents.

Figure 6 shows the current TLO design protocol. The planning agent is selected using a shared convention based on the MLO agents' names. No communication is needed for this. In the hierarchical MLO currently in use, the planner then asks its peers, via a `controlled-capabilities?` message, for information about their capabilities or capabilities of VIPs they control that would be useful for the current mission. When the planner receives the replies, it assigns VIPs to mission tasks based on their capabilities using the constraint-based task assignment technique.

The planner next creates a management structure for the task assignment, thus completing the hierarchical TLO. This is done using a simple best-fit strategy that chooses managers based on such heuristics as: first pick the manager with the smallest number of management capabilities; and if a potential manager is also one of the VIPs already assigned to a task, then prefer it as the manager for the task.

During this phase in the example AOSN, Ariel is selected by shared convention to be the planning agent. Figure 7 shows the hierarchy that it designs for the example mission. Note that a new task, `STN0`, is created as part of the design of the management structure. This is because no single VIP has sufficient resources to manage all the tasks. Consequently, this "middle manager" role is created and filled.

3.4 TLO Adjustment and Reorganization

Since it operates in the real world, an AOSN's situation will often significantly change during its operation, forcing it to change in response. This section focuses on protocols involved in handling two such changes when a TLO is in control: when a VIP exits the system, and when a new VIP enters. Protocols for entry

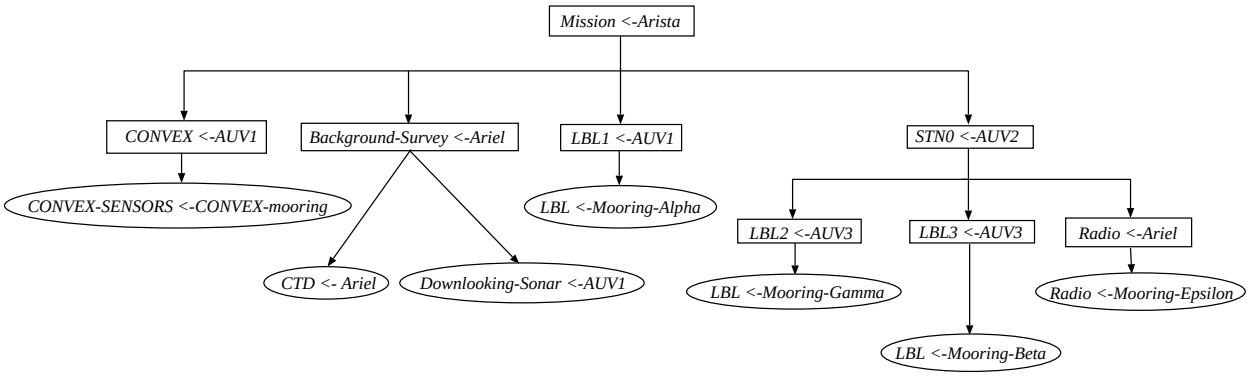

Figure 7: The TLO created for the example mission. Boxes represent tasks and are labeled with "taskname <- manager"; ovals represent atomic subtasks requiring one capability and are labeled "capability needed <- VIP assigned".

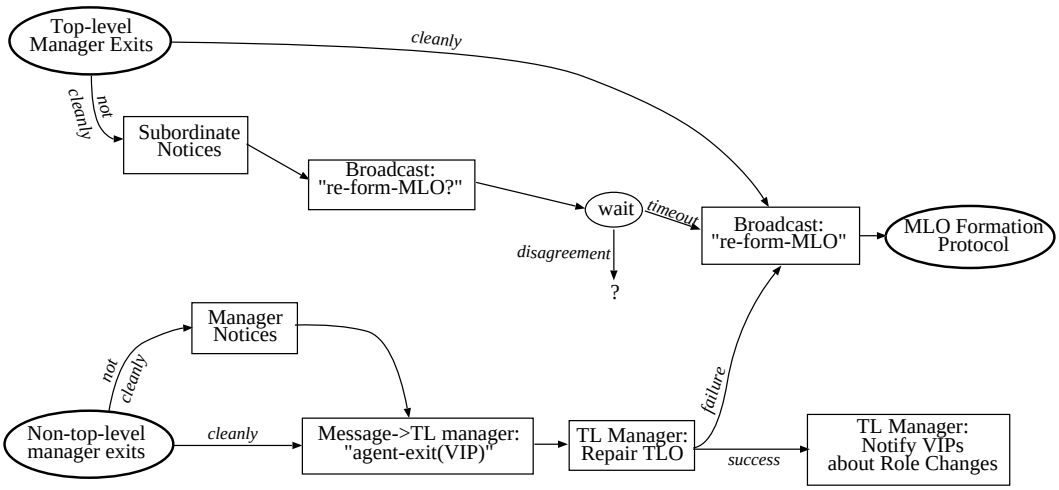

Figure 8: Protocol for VIP exit during TLO work phase.

and exit of VIPs during other phases exist, but are not discussed here in the interest of brevity. Protocols for other kinds of changes, such as TLO errors or changes in the mission or environment, will be the subject of future work.

3.4.1 VIP Exit Protocol

Figure 8 shows the protocol for handling the exit—or failure—of a VIP. There are four cases defined by two factors: whether the VIP is the top-level manager and whether the exit is abrupt or clean. By "clean", we mean that the VIP has sufficient time before exiting to follow the protocol, thus making the change easier for the TLO to handle. If the exit is abrupt, then the TLO has to detect somehow that the VIP has exited.

The easiest case is when the exiting VIP is the TLO's top-level manager, and it has time to exit cleanly. This VIP is in the best position to control what happens as a result.[3] In this case, the VIP broadcasts a `re-form-MLO` message to trigger any MLO agents present to begin re-forming an MLO to reorganize the AOSN. In the future, we will explore the possibility of the manager having enough information and time to replace itself from among the other VIPs present.

If the TLO's top-level manager exits abruptly, then one of its subordinates must notice that it is gone. This might happen, for example, when the manager fails to respond as expected to a message. For the current purposes, we assume that one of the MLO agents in the TLO can detect the problem and leave the exact mechanism for future work. The detecting agent broadcasts a `re-form-MLO?` message to ask any other MLO agents in the system their opinions about re-forming the MLO, then it waits. After a period of time, if there are no dissenting messages in the reply, it broadcasts a `re-form-MLO` message to initiate MLO re-formation. As shown in the figure, the case where there are disagreements is not yet handled.

If the exiting VIP is not the top-level manager, and it has time to exit cleanly, it sends a `agent-exit` message to the top-level manager. The manager then tries to replace it using any slack resources (unused capabilities of other VIPs) the AOSN has. If the damage to the TLO can be repaired this way, then the affected VIPs are told their new roles and work on the mission continues. If not, then the manager triggers the re-formation of the MLO via a `re-form-MLO` message.

If a VIP cannot exit cleanly, then much of the same sequence of events occurs, but only after one of the VIP's managers notices the problem.[4]

The process of re-forming the MLO is the same as that of forming an MLO when the AOSN is initially deployed. This allows any changes in the composition of the AOSN since deployment (or last re-formation)

[3]The protocol obviously depends on the TLO's organizational structure; this version is for hierarchies. Future work will address variants for other organizational structures.

[4]In the current implementation, a VIP may have more than one manager for the different tasks it is performing.

```
00:16:40.0 (SIM) ********************************
00:16:40.0 (SIM) **Event: (agent-exit mooring-Epsilon).
00:16:40.0 (SIM) ********************************
00:16:51.01 (TLO) Top-level manager (EAVE-Arista) received message from
            EAVE-Ariel that mooring-Epsilon has exited; attempting to repair TLO.
00:16:51.01 Attempting to repair TLO after exit of mooring-Epsilon.
00:16:51.01 Affected roles:
00:16:51.01   Labor role LABOR-ROLE6 (mooring-Epsilon using radio (1 unit) for
            radio)
00:16:51.01 Building repair problem REPAIR0.
00:16:51.01 Agent assignment successful.
00:16:51.01 Assigning mooring-Delta to LABOR-ROLE6, using capability radio for
            task RADIO.
00:16:51.01 Repair successful; TLO updated with fix.
00:16:51.01 (TLO) EAVE-Arista -> mooring-Delta: you now fill role LABOR-ROLE6.
00:16:51.01 (TLO) EAVE-Arista -> EAVE-Ariel: mooring-Delta now fills role
            LABOR-ROLE6, which you manage.
00:16:51.01 (TLO) EAVE-Arista: status of repair of exit of mooring-Epsilon is
            SUCCESS.
00:16:51.01 (TLO) Repair of mooring-Epsilon exiting complete.
00:16:52.02 (TLO) EAVE-Ariel -> mooring-Delta: start role LABOR-ROLE6.
00:16:53.06 (TLO) mooring-Delta: beginning work on LABOR-ROLE6.
```

Figure 9: Simulator output during TLO repair.

to be taken into account. In future work, we will investigate the benefits of allowing the MLO to remain active in some form while the TLO is working. This could shorten the time necessary to reorganize, although at the expense of the communication bandwidth and computational effort needed to maintain the MLO.

To demonstrate the protocols in action, suppose that in our example AOSN one of the moorings, Epsilon, exits abruptly. Its manager, Ariel (see Figure 7), notices this and informs the top-level manager, Arista. Arista then reassigns mooring Delta, which has the slack capability "radio", to the task, and the TLO continues carrying out the mission. Simulator output for this is shown in Figure 9.

In contrast, suppose that some time after a new MLO agent, Tenellia, has entered the AOSN, Ariel itself exits cleanly. In this case, the TLO does not have enough slack resources to repair itself. As shown in Figure 10, in this case the TLO dissolves, a new MLO forms, and a new TLO is created making use of the new MLO agent.

3.4.2 VIP Entry Protocol

New agents can enter the AOSN at any time. During some phases, any information about the new agent can and should be used immediately. For example, if an MLO agent arrives during MLO formation, it can immediately participate in the MLO; similarly, if any VIP arrives during MLO discovery, information about its capabilities should be integrated into what the MLO already knows about the AOSN's capabilities. In other cases, information about the new agent should be noted, even if it is not immediately useful. During the TLO work phase, for example, a new VIP's capabilities should be noted as slack resources, which might be used later for AOSN repair.

Figure 11 shows the protocol an agent follows when entering the AOSN. Also shown are the interactions of that protocol with the portion of the TLO work phase protocol dealing with the arrival of new agents. An entering MLO agent broadcasts a `organization-present?` message, and a non-MLO agent broadcasts a `VIP-organization-present?` message.

If the VIP is an MLO agent and there is no organization present, then the MLO formation protocol is followed from here. If there is an active TLO, then one or more of the MLO agents in that organization hearing the entry message will respond to any kind of entering VIP with a `existing-TLO-identify-yourself` message that includes information about who the top-level manager is. The entering VIP will then send a `VIP-info` message to the manager to tell it who it is, where it is, and what its resources and capabilities are. It then waits for the manager to tell it what to do. The manager adds the information about the VIP

```
[...new agent Tenellia has entered, capabilities include CTD and side-scan sonar...]
02:46:40.0 (SIM) ********************************
02:46:40.0 (SIM) **Event: (agent-exit EAVE-Ariel).
02:46:40.0 (SIM) ********************************
02:46:51.01 (TLO) Top-level manager (EAVE-Arista) received message from
            EAVE-Ariel that EAVE-Ariel has exited; attempting to repair TLO.
02:46:51.01 Attempting to repair TLO after exit of EAVE-Ariel.
02:46:51.01 Affected roles:
02:46:51.01   Task manager role TASK-MANAGER1 (EAVE-Ariel managing background-survey)
02:46:51.01   Labor role LABOR-ROLE1 (EAVE-Ariel using CTD (0 units) for
            background-survey)
02:46:51.01   Task manager role TASK-MANAGER5 (EAVE-Ariel managing radio)
02:46:51.01 Building repair problem REPAIR1.
02:46:51.01 Can't assign agents to repair TLO.
02:46:51.01 (TLO) EAVE-Arista: status of repair of exit of EAVE-Ariel is FAILURE.
02:46:51.01 (TLO) ** Cannot handle problem within TLO; gen1118 reinvoking MLO **
02:46:51.01 (TLO) EAVE-Arista->all: re-form-MLO
...
02:47:21.01 (MLO) EAVE-Arista: initiating MLO formation with agents =
            (EAVE-Arista)
...
02:47:51.01 (MLO) EAVE-Arista is attempting to discover other VIPs.
...
02:48:53.0 (MLO) Selecting EAVE-Arista as planner (convention: first in MLO).
...
02:49:53.0 (MLO) TLO created.
02:49:53.0 (MLO) EAVE-Arista: Telling TLO top-level manager Tenellia to begin work
02:49:53.0 (MLO) EAVE-Arista -> all: dissolve MLO.
02:49:53.0 (SIM) switching context -> TLO-work.
```

Figure 10: Simulator output during reorganization.

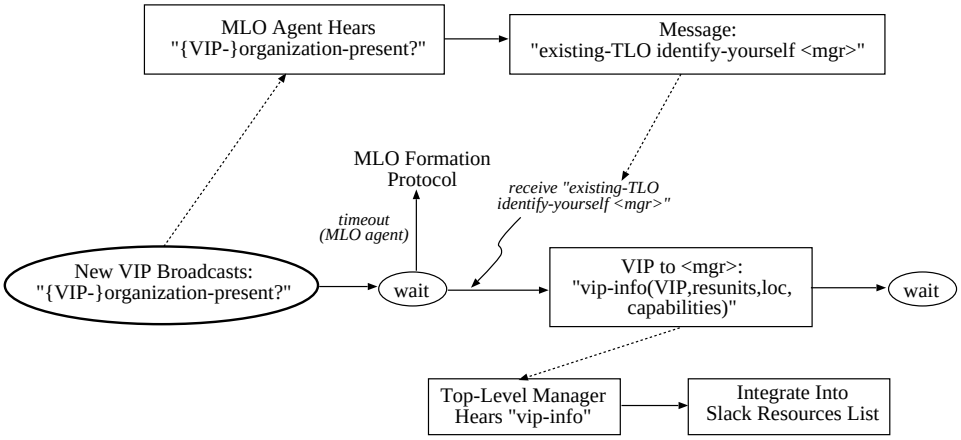

Figure 11: An agent entry protocol.

to its slack resources list.

When the TLO is working on a mission and a new agent enters, the new capabilities might allow the TLO to better accomplish the mission. The TLO should change to reflect this, or perhaps the AOSN should even reorganize. Recognizing such opportunities in a distributed system is the focus of a related project and is beyond the scope of this paper.

4 Simulation Results

In order to test performance of the organization/reorganization protocols, two sets of runs were performed, one with no agent exits/failures and one with agents exiting at different points during the mission. We refer to the former as the normal runs and the latter as the error runs. An experiment controller, written in Allegro Common Lisp, generated problems of various degrees of difficulty and AOSNs with varying properties and gave them to the CoDA simulator. Data of primary interest were the time needed to create a TLO and the communication bandwidth needed. In addition, the duration and required bandwidth of each phase (MLO formation, etc.) were measured. For the error runs, we were also interested in time and bandwidth performance during TLO repair and AOSN reorganization, as well as whether or not repair/reorganization was successful. In both cases, CLIP (Common Lisp Instrumentation Package) and CLASP (Common Lisp Analytical Package) [12] were used to gather and analyze data. In addition, Microsoft Excel was also used to analyze the data.

It should be stressed that the failure rate of the VIPs in these runs is in no way meant to approximate what a real AOSN would encounter in practice. We would expect the real failure rate usually to be significantly lower. The failure rates used in these experiments give an approximation of the worst-case performance of the protocols.

4.1 Runs with No Agent Exits

For the normal runs, the total number of VIPs varied from 10 to 15, with the number of MLO agents present varying from 2 to 4. The number of tasks varied from 7 to 10, the capabilities per VIP from 3 to 7, and the number of agents that could manage others from 3 to 7. The total number of capabilities from which a VIP's capabilities were drawn was 15, and the number of other agents a manager could manage was set to 6. Agents were allowed to enter the system initially over a period of 35 seconds, simulating the near-simultaneous initialization of agents in an AOSN. For these parameters, there were 761 runs out of 1800 in which a TLO could be formed, given the capabilities present and the tasks to be assigned. This high failure rate is an artifact of the simulation process' random generation of AOSNs and missions. We would expect real AOSNs to be fielded with some attention to providing sufficient resources and capabilities for the range of missions they would be expected to encounter over their lifetimes. Changes in the composition of the AOSN and the assignment of missions beyond the usual range would, of course, cause failures, but we would expect the failure rate to be significantly lower than for these randomly-constructed AOSNs and missions.

The mean time to form a TLO for successful cases was 196.6 seconds (std. dev. 7.5). Bandwidth can be estimated in several ways. The mean number of point-to-point messages transmitted was 64.2 (std. dev. 7.1), and the mean number of broadcasts was 35.8 (std. dev. 4.7). The mean number of symbols transmitted was 1130.6 (std. dev. 143.7). The mean number of symbols per second, a better measure of the true bandwidth, was 5.8 (std. dev. 0.8).[5]

The overall time to TLO formation is remarkably robust to changes in the number of MLO agents or the total number of VIPs. This is in large part due to the various time-out periods used in the protocols, which are much longer than the typical transit time for a message via the (simulated) acoustic modems.

The time until TLO was formed was affected by number of MLO agents ($p<0.001$, two-factor ANOVA without replication) but not by number of VIPs. Time to TLO formation *decreased* linearly with the number of MLO agents ($R^2=0.972$). Perhaps an increased number of MLO agents reduces the time any particular agent spends waiting for messages from its controlled VIPs (during MLO discovery), since there will be on average fewer VIPs per MLO agent. The effect is slight, however, and we will examine this in more detail with a wider range of MLO agents in the future.

Bandwidth, estimated by symbols transmitted per second, was affected by both the number of MLO agents and the total number of VIPs ($p<.001$, two-factor ANOVA without replication). It increases linearly with the number of MLO agents ($R^2=0.989$) and also linearly with the number of VIP agents ($R^2=0.998$). This is encouraging, as it suggests that the protocols will gracefully scale up to large numbers of VIPs.

[5] Actual bandwidth, in bits per second, was not measured, since it depends on the agent communication language as well as the amount of error correction used.

4.2 Runs in Which Agents Exit

To test the effect of agent exits, including agent failures, 768 runs were conducted with either 3 or 4 MLO agents, 9 or 12 VIPs total, 10 or 12 tasks, 3, 5, or 7 capabilities/VIP, 8 management capabilities per manager, and either 5 or 7 managers. Agents were allowed to leave in any phase, and their exit could be either clean or abrupt, as defined above. Of these runs, 371 resulted in failure, that is, the MLO was unable to create a TLO. In 120 runs, repairs were attempted after a TLO was formed, and, of these, 84 resulted in success: 71 in which the TLO was able to adjust itself to the loss of an agent, and 13 in which a complete reorganization, mediated by an MLO, was needed. In 36 cases, repair could not be carried out with the capabilities remaining in the AOSN after the agent exited.

Repairs and reorganizations only occur due to an agent exiting during the TLO work phase. Our simulator is not currently instrumented to directly measure the impact of agent exits in other phases. However, we can compare the overall failure rate between the normal runs, in which there were no agent exits, and these error runs. The normal runs had 761 successes out of 1800 tries, or 42.3%. The error runs had 397 successes out of 768 tries, or 51.7%. Since the parameters were different between the two kinds of runs, we cannot infer too much from this. However, it at least leads us to believe that agents exiting in the early phases of operation do not have a large effect on the overall success rate of the CoDA protocols, and hence that the protocols are robust to agent exits, including agent failures. This will be tested experimentally in the future.

With respect to the effect of agent exit on time to TLO completion, for the 397 successful runs in this series, the mean time was 200.3 seconds (std. dev. 31.1). Means for messages and broadcasts were 66.1 (std. dev. 15.3) and 34.4 (std. dev. 7.3), respectively, and for total symbols transmitted, 1178.0 (std. dev 242.4). The mean bandwidth, estimated by symbols/second, was 5.9 (std. dev. 1.0). Again, since the runs had different parameters, statistical tests can tell us little, yet it seems that the overhead of handling agent exits was slight, as measured by time to TLO formation and communication bandwidth needed.

5 Related Work

Most of the research in ocean engineering on controlling multiple AUVs has taken place in the context of systems that are much simpler than the kinds of AOSNs we are concerned with. For example, one of the earliest projects in this area was MAUV (Multiple AUV) [13], a direct ancestor of the CoDA Project. MAUV focused on simple, hierarchical multi-AUV systems. The project developed mechanisms, based on the NASREM control architecture [14], to control two EAVE [9] AUVs during in-water tests. A later project along these lines was MAVIS (Multiple Autonomous Vehicle Imaging System) [15], again focusing on cooperation between two EAVE vehicles. In both of these projects, the number of vehicles was small, the tasks were simple, and the system was not considered to be open: that is, failure of an AUV would terminate that run of the system.

Other, similar systems have since been deployed, including multi-vehicle tests by researchers at Florida Atlantic University [16] as well as at MIT Sea Grant as part of their AOSN MURI project [3; 17; 2]. These systems, though serving as critical testbeds for AOSN technology, have not resulted in the development of AOSN control mechanisms sufficient for the kinds of AOSNs we are interested in: ones that are open, heterogeneous, long-duration, and flexible.

There has been increasing attention in recent years in using multiple AUVs as mine countermeasure systems [e.g., 2]. These have focused mainly on homogeneous systems of inexpensive vehicles (e.g., REMUS [18]), with relatively simple control mechanisms for the fleet of vehicles. Although such systems are open due to the nature of the task, they are simpler than a general-purpose AOSN. They are homogeneous, so replacement of a lost vehicle is simpler; and they are focused on one task, so their control software can be simpler.

The Ocean SAMON (SAmpling MObile Network) Controller project [19] addresses some of the same problems as CoDA. SAMON is a control mechanism for an AOSN that can self-organize and reorganize, to a limited extent, when something goes wrong. SAMON deals with the open system issue, and it can be re-tasked after deployment. SAMON utilizes finite state machines to model protocol-like interactions between agents and is thus similar in some regards to CoDA. Early descriptions of SAMON indicated

that it was designed for a system of homogeneous agents [19]. Apart from some fixed sensor packages with which the AOSN work area is seeded, all SAMON agents were AUVs that can assume each other's functionality, including supervisory control of other agents, as the need arises. This makes SAMON's control task easier than CoDA's, but it places limits on the kinds of VIPs that can participate in an AOSN controlled by SAMON. However, a more recent report indicates that SAMON has added some ability to coordinate heterogeneous AUVs via using a common command language [20]. In addition, SAMON is meant to operate in conjunction with a "tactical coordinator", possibly a human, that is remote from the AOSN work site, whereas CoDA is designed to be autonomous if need be. SAMON's reorganization ability is somewhat limited, as well. When a supervisory AUV (SAUV) fails, one of the other AUVs can be "promoted" to supervisory status, and areas of responsibility can be redrawn in response to changes in the situation. In contrast, CoDA's MLO can completely re-design the task-level organization in response to changes, and in the future, CoDA will allow a variety of organizational structures. However, CoDA could benefit from the kind of careful mathematical modeling of behavior that the SAMON researchers have done in their work, although not necessarily with a basis in finite state machines.

There has been a great deal of work in the multiagent system (MAS) and cooperative distributed problem solving (CDPS) communities on cooperation between autonomous and semi-autonomous agents. One of the earliest approaches was multiagent planning [e.g., 21]. In this approach, a single agent creates a plan incorporating all actions to be taken by all agents. This is good from the standpoint of creating an effective plan, but it requires global knowledge down to the level of actions taken by individual agents. In an AOSN, some global knowledge will likely be available or obtainable, such as which agents are present, what their capabilities are, what the mission is, and some general information about the environment. The level of knowledge needed by multiagent planning, however, may in general be difficult or impossible to obtain, and it may be too much to ask of any one agent to process such a large amount of knowledge. The chief disadvantage of multiagent planning is the fact that there is a single point of failure, the planner. Should the planner fail, the system as a whole may fail.

One of the most successful approaches to multiagent coordination has been work growing out of the Contract Net Protocol (CNP) [22]. CNP uses a market metaphor for structuring organizations of agents in a truly distributed fashion: when an agent has a task to be done, it requests bids from other agents, then awards a contract to the winner. The result is a dynamically-created, task-specific hierarchy. CNP can be used with heterogeneous agents, and it requires an agent to have minimal knowledge of other agents. It has been extended in many ways to take into account prices and coalitions of agents [e.g., 23].

Two major problems arise when considering using CNP for AOSN control. First, there is no global perspective in CNP, and consequently, no guarantee that the organization formed will be a good one for the task and situation. The organization emerges from the pattern of contracts; it is not designed based on any global knowledge, as is the case when CoDA's MLO designs a TLO. Second, CNP is not particularly robust to failures of its agents. Failure handling was not part of the original protocols and has received little attention since. Due to the lack of global control, any repairs that are effected will be done solely from a local perspective. Consequently, the system will miss opportunities to reorganize when that would be best. Finally, a hierarchy may not be the best organization for all tasks, yet that is the only organization that CNP provides.

One of the most flexible approaches to CDPS is the Functionally Accurate/Cooperative (FA/C) family of coordination algorithms [24]. The idea behind this approach is that useful coordination can occur even in the absence of global information by allowing agents to form locally-coherent plans that include the actions of their neighbors. Even though the resulting set of plans may not be globally-coherent, it will be satisficing, or functionally accurate. The earliest of these algorithms was Partial Global Planning (PGP) [25], and more recently this has been generalized to create Generalized Partial Global Planning (GPGP) [26; 27].

FA/C algorithms are attractive from the standpoint of self-organization, since partial global plans emerge from local interactions between agents. However, as with CNP, there is no global perspective, which is critical for AOSN control to ensure that mission goals are achieved such that the scientist/user's requirements for the data are met. The organization growing from (G)PGP, essentially a shared plan with agents agreeing to their roles within it, may not be optimal for the situation; local criteria for data quality or goal satisfaction may be met, yet global criteria, which are not taken into account, may fail. Consequently, the organization

of the agents, if not the particular actions to be taken in the context of the organization, should be designed, not allowed to emerge.

Another disadvantage of FA/C from our perspective is that it requires sophisticated planning algorithms to exist on all of the agents, and it requires the exchange of a significant amount of data in the form of individual and partial plans. Though the agents need not be homogeneous with respect to their sensors, etc., they need to be nearly homogeneous with respect to their level and type of intelligence.

STEAM (Shell for TEAMwork) [28] is concerned with allowing a group of agents to function as a team. There exist team goals, mutual beliefs, and joint commitments between the agents. STEAM provides team synchronization, plan monitoring and repair, and decision-theoretic communication selectivity. Agents use a model of teamwork to reason about mutual beliefs, their roles in the team, and when to communicate. There are team operators, initiated by the team leader, that must be simultaneously selected by the agents involved to achieve shared team goals. The approach has been tested in the domain of helicopter attack/transport scenarios and Robosoccer.

A problem with this approach is that the team is the only organizational structure that it uses, which may not be appropriate for all situations and missions. As with the FA/C approach, it also requires all agents to engage in sophisticated reasoning, which limits the kinds of VIPs that could participate in a system controlled by STEAM.

Barbuceanu [29] has developed an approach to multiagent coordination based on viewing organizations as systems that constrain their members by means of mutual obligations and interdictions defined by the roles the agents play in the organization. A benefit of such a view over, say, requiring that there be mutual goals for cooperation, is that it can be used in systems where the agents may not have mutual goals, may not know about them, or may not have the processing power to reason about such goals. However, for AOSNs, we can assume mutual goals exist, and in order to assure the timeliness and quality of data samples, it will be advantageous to allow organizations in which direct commands control subordinates' behavior rather than allowing agents the latitude provided by obligations and interdictions. In addition, this approach requires all of the agents to be intelligent enough to understand the interdictions and obligations and how they affect their behavior.

There are also behavior-based approaches to collective behavior that take a more bottom-up approach to coordination. For example, one approach suggests *basic behaviors* as the basis for intelligent collective behavior arising from local interactions [30]. Composing these basic behaviors results in higher-level group behavior (e.g., flocking) emerging. Though interesting and potentially useful for some aspects of AOSN tasks (e.g., following an ocean front), these behavior-based approaches cannot guarantee that the AOSN's global goals are satisfied.

Some work has been done on organizational self-design [e.g., 31], that is, allowing organizations of autonomous agents to self-organize and reorganize based on knowledge about organizations. For example, Gasser & Ishida [31] have developed an agent "microarchitecture" to allow agents themselves—packets of rules, essentially—to split and join to change the organization they are in. While this is not directly applicable to the type of agent present in AOSNs, more recent work has begun to look at designing tree-structured organizations based on features of the task and environment [32]. We anticipate making use of some aspects of this work in future versions of CoDA.

6 Conclusion

Controlling advanced AOSNs is a challenging problem. The AOSN must be autonomous, it will be heterogeneous, and it has no choice but to be open, with VIPs coming and going during its operation. Consequently, it must be able to organize its components into an effective data-gathering system and reorganize itself as the environment, the mission, or its composition changes.

In this paper, we have presented CoDA, a two-level, protocol-based approach to controlling advanced AOSNs. A two-level approach is appropriate because it allows us to avoid the flexibility/efficiency trade-off. The MLO is very flexible and places few requirements on the composition of the AOSN in order to form and carry out its duties. The TLO it creates, in contrast, is not very flexible, but is highly-tailored to the current situation, allowing efficient use to be made of the AOSN's resources.

Protocols are the right approach, we believe. They allow a heterogeneous group of agents to cooperate without needing to know anything about the others other than their capabilities and that they will follow the protocols. Protocols are easily simulated and foster rapid prototyping of AOSN control mechanisms. Protocols also promote generality, allowing details of AOSNs and mission classes to be ignored during initial design of the approach, while allowing the details to be taken into account as the protocols are tuned to the particular AOSN and class of missions they will be used for.

Simulation results indicate that the protocols work for complicated AOSNs and missions. Self-organization into an MLO is possible as long as there is at least one MLO agent present, and the protocols lead to successful TLO formation when there are sufficient capabilities present in the AOSN. Both organization time and the bandwidth used by the agents while following the protocols grow linearly with the number of MLO agents and VIPs in the system, which bodes well for scaling up to larger AOSNs. The CoDA protocols handle well agent failure in all phases without adding much overhead in terms of time or bandwidth utilized.

Future work will focus on further developing and refining the protocols. This will be done in part by increasing the fidelity of the simulation as we work toward eventual in-water tests of the CoDA protocols. With more realistic simulations, assumptions underlying the current protocols will be exposed and can be addressed to give rise to future versions of the protocols. For example, we will need to simulate different kinds of acoustic networks, different classes of VIPs, and so forth. In support of this, we intend to connect our CoDA simulator to the CADCON high-fidelity multiagent simulator [33] and other simulators to allow the protocols to be used to control highly-realistic simulated agents. Some work on this has already been done that will facilitate testing CoDA with other laboratories' high-fidelity simulators as well as with our own land robots [34].

Work on the protocols has to date focused on organization and reorganization of the AOSN, with little attention paid so far to coordination of the agents during the task-level organization's work on the mission. Future work will focus on this aspect of the CoDA protocols. This will draw on the large body of literature in distributed artificial intelligence, multiagent systems, and distributed systems that focuses on multiagent coordination problems. This work will be critical to support more realistic simulation experiments as well as actually fielding the CoDA approach.

The current CoDA simulator combines several simulation methodologies, including discrete event simulation, but it does not have a well-developed mathematical model of the process of agents following the protocols. Results have therefore so far been empirical rather than analytical. One promising direction for developing a mathematical model is to represent agents following the protocols by a Petri net (e.g., a timed, Colored Petri net [35]). Such networks have been used to verify such properties of distributed systems as reachability of states and freedom from deadlock. We anticipate that developing such a model of the CoDA protocols will be an area of future work.

Future work will also involve continuing to develop mechanisms for individual agents to use to enable them to follow the protocols. This ties in with our other projects in agent-level control and interagent communication, as well as the other parts of the CoDA project focusing on task assignment and organization selection/design.

Future versions of the protocols, as well as other parts of the CoDA project, will be tested in simulation experiments designed to characterize their behavior as well as compare them to other approaches. The eventual goal is to test our work aboard actual VIPs in a fielded advanced AOSN.

Acknowledgments

The authors would like to thank the other members of MaineSAIL (University of Maine Software Agents and Artificial Intelligence Laboratory) for their insightful comments and other help over the course of this work, as well as our collaborators at the Autonomous Underwater Systems Institute (AUSI). We are deeply grateful to the Office of Naval Research for their generous support of this work through grants N0001-14-96-1-5009 and N0001-14-98-1-0648. The content does not necessarily reflect the position or the policy of the U.S. government, and no official endorsement should be inferred. Thanks also to the anonymous reviewers for a very careful reading of the manuscript and many helpful comments.

References

[1] T. Curtin, J. Bellingham, J. Catipovic, and D. Webb. Autonomous oceanographic sampling networks. *Oceanography*, 6(3), 1993.

[2] B. A. Moran, R. J. Grieve, and H. Schmidt. GOATS'98–AUV networks sonar concepts for shallow water mine countermeasures. In *Proceedings of the Eleventh International Symposium on Unmanned Untethered Submersible Technology*, pages 121–132, Durham, NH, August 1999.

[3] J. G. Bellingham, H. Schmidt, and C. Chryssostomidis. AOSN MURI: Real-time oceanography with autonomous ocean sampling networks: A center for excellence. On the World Wide Web at `http://auvserv.mit.edu/MURI/annual-report98.htm`, accessed 12/30/99, last modified 12/31/98., 1998. (MIT AOSN MURI Annual Report).

[4] C. Hewitt. Offices are open systems. *Communications of the ACM*, 4(3):271–287, 1986.

[5] R. M. Turner and E. H. Turner. Organization and reorganization of autonomous oceanographic sampling networks. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation (ICRA'98)*, pages 2060–2067, Leuven, Belgium, May 1998.

[6] R. Turner, E. Turner, and D. Blidberg. Organization and reorganization of autonomous oceanographic sampling networks. In *Proceedings of the 1996 IEEE Symposium on Autonomous Underwater Vehicle Technology*, pages 407–413, Monterey, CA, June 1996.

[7] E. H. Turner and R. M. Turner. A constraint-based approach to assigning system components to tasks. *Applied Intelligence*, 10(2/3):155–172, 1999.

[8] F. Bub, W. Brown, P. Mupparapu, K. Jacobs, and B. Rogers. Hydrographics survey report: Convective overturn experiment (CONVEX): R/V *endeavor* cruise en–291. Technical report, University of New Hampshire Ocean Process Analysis Laboratory, 1997. (WWW: `http://ekman.sr.unh.edu/OPAL/CONVEX/EN291/en291_report.html`).

[9] D. R. Blidberg and S. G. Chappell. Guidance and control architecture for the EAVE vehicle. *IEEE Journal of Oceanic Engineering*, OE–11(4):449–461, 1986.

[10] R. M. Turner, D. R. Blidberg, S. G. Chappell, and J. C. Jalbert. Generic behaviors: An approach to modularity in intelligent systems control. In *Proceedings of the 8th International Symposium on Unmanned Untethered Submersible Technology (AUV'93)*, Durham, New Hampshire, 1993.

[11] K. Sycara, S. Roth, N. Sadeh, and M. Fox. Distributed constrained heuristic search. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1446–1461, 1991.

[12] S. D. Anderson, D. L. Westbrook, M. Schmill, A. Carlson, D. M. Hart, and P. R. Cohen. *Common Lisp Analytical Statistics Package: User Manual*. Department of Computer Science, University of Massachusetts, 1995.

[13] J. S. Albus. Multiple Autonomous Undersea Vehicles. Technical Report NIST Technical Note 1251, National Institute of Standards and Technology, Gaithersburg, MD 20899, 1988.

[14] J. S. Albus, H. G. McCain, and R. Lumia. NASA/NBS standard reference model for telerobot control system architecture (NASREM). Technical Report NIST Technical Note 1235, U.S. Department of Commerce, National Institute of Standards and Technology, NIST, Gaithersburg, MD 20899, 1989.

[15] R. M. Turner, D. R. Blidberg, J. S. Fox, and E. H. Turner. Multiple autonomous vehicle imaging system (MAVIS). In *Proceedings of the 7th International Symposium on Unmanned Untethered Underwater Submersible Technology (AUV '91)*, 1991.

[16] S. Smith, K. Ganesan, S. Dunn, and P. An. Strategies for simultaneous multiple AUV operation and control. In *IARP'96*, France, 1996.

[17] H. Schmidt, J. G. Bellingham, M. Johnson, D. Herold, D. M. Farmer, and R. Pawlowicz. Real-time frontal mapping with AUVs in a coastal environment. In *Proceedings of the IEEE Oceanic Engineering Society Conference (OCEANS'96 MTS)*, pages 1094–1098, 1996.

[18] R. Stokey, T. Austin, C. von Alt, M. Purcell, R. Goldsborough, N. Forrester, and B. Allen. AUV bloopers or why Murphy must have been an optimist: A practical look at achieving mission-level reliability in an AUV. In *Proceedings of the Eleventh International Symposium on Unmanned Untethered Submersible Technology*, pages 32–40, Durham, NH, August 1999.

[19] S. Phoha, J. Stover, R. Gibson, E. Peluso, and P. Stadter. Autonomous ocean sampling mobile network controller. In *Proceedings of the Tenth International Symposium on Unmanned Untethered Submersible Technology (UUST)*, pages 362–374, Durham, NH, September 1997.

[20] S. Phoha. A proposal for a generic behavior message-passing language for collaborative missions of heterogeneous autonomous underwater vehicles. Penn State University/ARL Memo to the AOSN Community, 1999.

[21] M. P. Georgeff. A theory of action for multi-agent planning. In *Proceedings AAAI-84*, pages 121–125, 1984.

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

[23] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 265–262, 1993.

[24] V. R. Lesser. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(6):1347–1362, November/December 1991.

[25] E. H. Durfee and V. R. Lesser. Using partial global plans to coordinate distributed problem solvers. In *Proceedings of the 1987 International Joint Conference on Artificial Intelligence*, pages 875–883, 1987.

[26] K. S. Decker and V. R. Lesser. Generalizing the Partial Global Planning algorithm. *International Journal on Intelligent Cooperative Information Systems*, 1(2):312–346, 1992.

[27] K. S. Decker and V. R. Lesser. Designing a family of coordination algorithms. In *Proceedings fo the First International Conference on Multi-Agent Systems (ICMAS)*, San Francisco, June 1995.

[28] M. Tambe. Agent architectures for flexible, practical teamwork. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 22–28, 1997.

[29] M. Barbuceanu. Coordinating agents by role-based social constraints and conversation plans. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 16–21, 1997.

[30] M. J. Matarić. Issues and approaches in the design of collective autonomous agents. *Robotics and Autonomous Systems*, 16:321–331, 1995.

[31] L. Gasser and T. Ishida. A dynamic organizational architecture for adaptive problem solving. In *Proceedings of the National Conference on Artificial Intelligence*, pages 185–190, July 1991.

[32] Y. pa So and E. H. Durfee. Designing tree-structured organizations for computational agents. *Computational and Mathematical Organization Theory*, 2(3):219–246, 1996.

[33] S. G. Chappell, R. J. Komerska, L. Peng, and Y. Lu. Cooperative AUV Development Concept (CADCON) – an environment for high-level multiple AUV simulation. In *Proceedings of the 11th International Symposium on Unmanned Untethered Submersible Technology (UUST99)*, Durham, NH, August 1999. The Autonomous Undersea Systems Institute, Lee, NH.

[34] R. Turner and J. Mailman. Interfacing the Orca AUV controller to the NPS UVW and to a land robot. In *Proceedings of the 11th International Symposium on Unmanned Untethered Submersible Technology (UUST)*, Durham, NH, August 1999.

[35] M. Reid and W. Zuberek. Timed Petri net models of ATM LANs. In J. Billington, M. Diaz, and G. Rozenberg, editors, *Application of Petri Nets to Communication Networks*, number 1605 in Lecture Notes in Computer Science, pages 150–175. Springer, 1999.