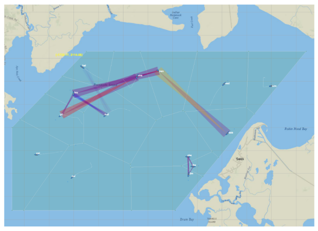
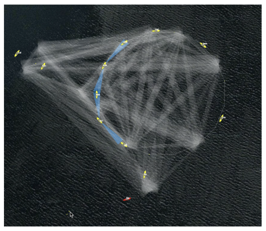
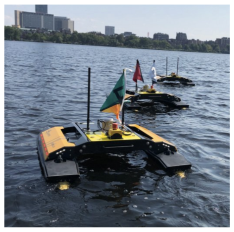
Decentralized Decision Making

Nov 2023

Michael Benjamin, mikerb@mit.edu
Department of Mechanical Engineering
MIT, Cambridge MA 02139

`project-pavlab/memos/memo_swarm_decentral`

1 Decentralized Decision-Making

The goal of decentralized decision-making is to allow a group of vehicles to be tasked with a specific goal, i.e., task, with the expectation that the vehicles will reach a consensus among themselves without involving further human input. In some cases this can be accomplished by a shared protocol where it is always clear what each vehicle should do, presuming there is some minimal awareness of ownship state relative to its collaborators, such as vehicle position. However, in other cases, even when ownship and collaborator vehicle positions are known, a dialog between vehicles is required to reach a consensus. Auction-based consensus algorithms are not new, but the method and implementation of this approached, integrated into an existing behavior-based architecture with dynamic behavior spawning is the focus of this portion of the work.

1.1 Static and Dynamic Behavior Configuration

Helm behaviors, including the Convoy behavior, are configured in part through static parameters specified at the time of mission launch, and dynamic updates to parameters provided at run-time, as depicted in Figure 1.

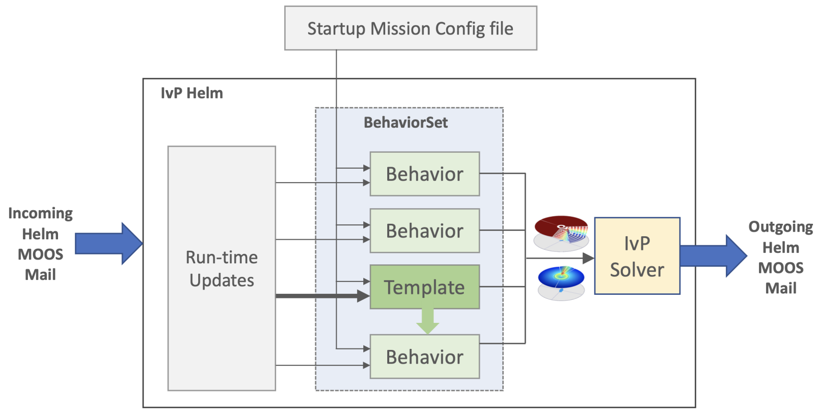


Figure 1: **Behavior Inputs**: Most behavior parameters are provided upon startup from the mission configuration file. Dynamic updates may be provided at run-time from any other MOOS application, allowing for higher levels of planning, and command and control to influence the autonomy as a whole through changes to the behaviors. Behavior *templates* are used in cases where behaviors come into existence only upon external events and incoming messages to the helm. A behavior template may spawn several instances during the lifetime of the mission.

Behaviors of the IvP Helm may be configured as *templates*, essentially starting the mission waiting for events to occur at run-time to spawn one or more instances of the templated behavior. Through this mechanism, the behavior configuration need not know the details of the tasks involved in the the eventual spawned behavior. This information will arrive as run-time updates, spawning behaviors as needed. Once the behaviors have achieved their objectives, the spawned behaviors will be deleted and removed from helm memory.

The support for dynamic behavior spawning, memory management, and post-mission analysis of dynamic behavior events, is a core (perhaps unique) strength of the helm and a key enabler for the family of swarm behaviors. The circumstances that govern dynamic behavior spawning, are handle through one (currently three) event managers.

1.2 Autonomy Event Managers

A key design goal is to minimize or eliminate any mission configuration as a mission varies in the number vehicles, vehicle ID names, or initial starting conditions. An operator should be able to power on any set of vehicles in any location or position and just say ”go”, where ”go” includes goals for the group to achieve. The operator may wish to later change the goal details or parameters, but the initial stage should amount to no more than a button-push.

A multi-vehicle collaborative mission utilizes up to three event managers available to the IvP Helm. The contact manager ingests position and pose node reports of other vehicles in the field. The obstacle manager ingests information about obstacles to be avoided. The task manager accepts tasking either from external command messages, or from collaborating vehicles.

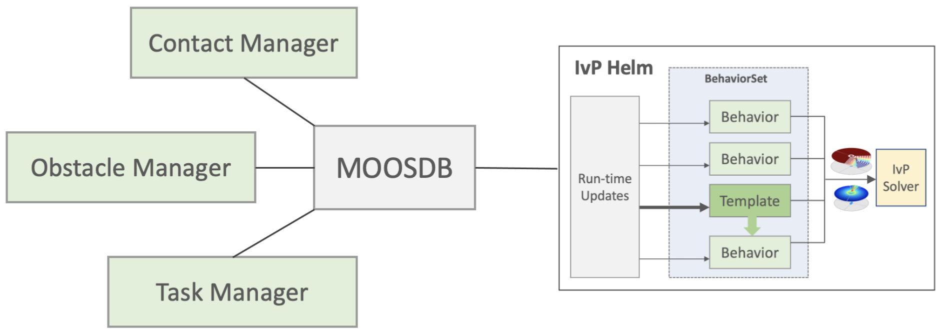


Figure 2: **Event Managers**: Event managers are MOOS apps that abstract a layer of complexity between communications and sensing systems and the behaviors of the helm. Behaviors register with the event managers for alerts. The registration specifies the conditions and format of the desired alerts. Alerts typically result in the spawning of new behaviors from behavior templates.

1.3 The Task Manager

The Task Manager coordinates incoming tasks and the spawning of proper behaviors of the helm. It is a thin application with no logic related to any particular task type. It is merely a broker to shield some complexity from the helm, and it also serves as a single entity to monitor overall task status on the vehicle across perhaps multiple tasks. The main loop is shown in Algorithm 1.

Algorithm 1: Task Manager Main Loop

```
1: procedure IterateZ
2:     handleNewMissionTasks()                    ▷ From operator or other apps
3:     handleNewAlertRequests()                   ▷ From Helm behavior(s)
4:     for each task do
5:         if task.state = tasked then
6:             postAlert(task)
7:             task.state ← alerted
8:         end if
9:     end for
10: end procedure
```

Typically tasks originate from one of two sources: either a direct command from an operator to the group of vehicles, or from a collaborating vehicle. In the case of the convoying mission, the initial task comes from the operator to choose a leader to lead the convoy. The lead vehicle then initiates a task to find a follower and so on until there are no vehicles remaining. The task manager begins a mission knowing nothing about any tasks.

When the helm launches, task behaviors will generate a message to the task manager to register for alerts when or if the task manager receives certain tasks. The auctioning between vehicles, and the auctioning algorithm for determining bids, is completely implemented in the task behaviors. General task behaviors are described next.

1.4 Task Behaviors

A task behavior is a special behavior in the IvP Helm. It does not generate an objective function to influence the vehicle trajectory. Its sole purpose is to participate in a task auction and, depending on the result, post output that may spawn a new behavior. A newly spawned behavior is the usual result for a task auction that has been won. Like any helm behavior though, many of the actions taken upon win or lose in an auction may be configured in the mission file and mission developers have wide latitude to improvise.

The main loop of a generic task behavior is shown in Algorithm 2 below, with much of the same logic depicted in Figure 3. A task behavior is meant to live briefly and progress through the following states: {*spawned, noroster, roster, bidding, bidwon, bidlost*}. The *spawned* state is the initial state upon spawning by the task manager. The task behavior's first job is to retrieve a list of teammates from the contact manager, a separate MOOS app. The request is shown on line 3 below. The contact manager maintains information about all contacts, but this request from a task behavior may filter the query for contacts that are of a particular type or range. Once this request is made, the behavior stays in the *noroster* state until the roster has been received.

Algorithm 2: Task Behavior Main Loop

```
 1: procedure OnRunState()
 2:     if state = spawned then
 3:         postContactReportRequest()                    ▷ For Contact Manager
 4:         state ← noroster
 5:     else if state = noroster then
 6:         if roster.received = true then
 7:             state ← roster
 8:         end if
 9:     else if state = roster then
10:         postBidToTeam().
11:         state ← bidding
12:     else if state = bidding then
13:         if allbids.received = true then
14:             if our_bid > highest_other_bid then
15:                 postBidWonFlags()
16:                 state ← bidwon
17:             else
18:                 postBidLostFlags()
19:                 state ← bidlost
20:             end if
21:         end if
22:     end if
23:     if state = bidwon or state = bidlost then
24:         initiateBehaviorCompletion()
25:     end if
26: end procedure
```

Once the task behavior has the roster of collaborators, it generates a bid for the current auction and sends the bid out to the vehicles on the roster. At this point it enters the *bidding* state and remains in this state until incoming bids have been received by every member of the roster. The function call to generate a bid, on line 10, is where the unique nature of the task behavior resides. Each task behavior of the IvP Helm shares the common structure of Algorithm 2, differing primarily by the implementation of bid generation and the registration of information needed for the bid generation.

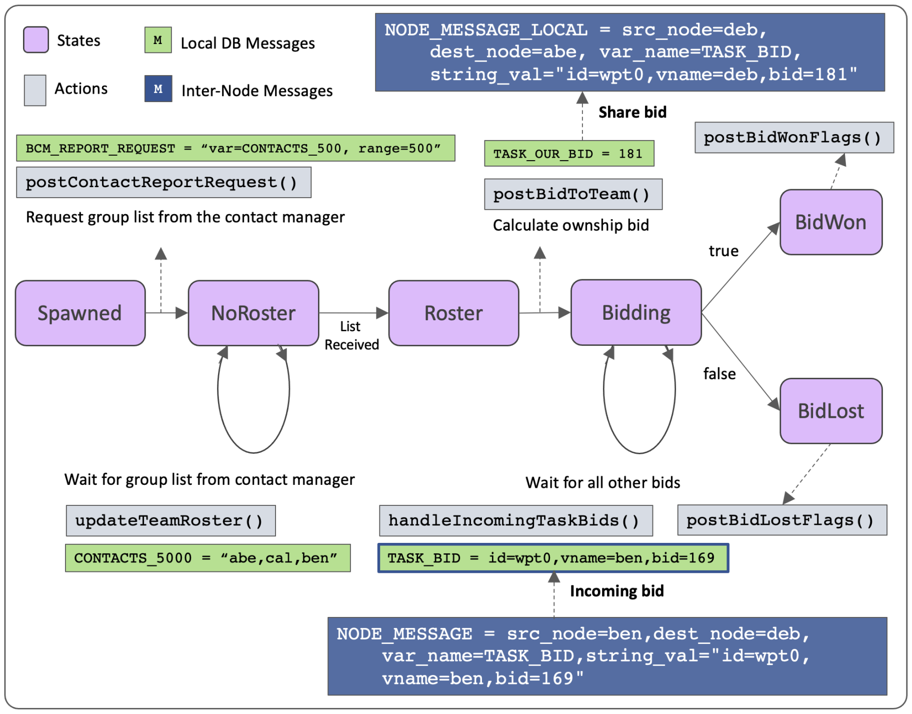

States
Local DB Messages
Actions
Inter-Node Messages
NODE_MESSAGE_LOCAL = src_node=deb, dest_node=abe, var_name=TASK_BID, string_val="id=wpt0,vname=deb,bid=181"
Share bid
BCM_REPORT_REQUEST = "var=CONTACTS_500, range=500"
postContactReportRequest()
Request group list from the contact manager
TASK_OUR_BID = 181
postBidToTeam()
Calculate ownship bid
postBidWonFlags()
BidWon
true
Spawned
NoRoster
List Received
Roster
Bidding
false
BidLost
Wait for group list from contact manager
Wait for all other bids
updateTeamRoster()
handleIncomingTaskBids()
postBidLostFlags()
CONTACTS_5000 = "abe,cal,ben"
TASK_BID = id=wpt0,vname=ben,bid=169
Incoming bid
NODE_MESSAGE = src_node=ben,dest_node=deb, var_name=TASK_BID,string_val="id=wpt0, vname=ben,bid=169"

Figure 3: **Task Behavior Progression**: A task behavior proceeds through the states shown, beginning from initial spawning by the task manager, and generation of bids to collaborators. Depending on ownship bid compared to collaborators, the task behavior will generate further publications, i.e., flags, that will spawn the appropriate behavior to perform the task. The primary difference between different task behaviors is in the implementation of bid generation.

1.5 Example Mission: Encircle and Defend High Value Asset

The first mission chosen to test and demonstrate the decentralized auction algorithms involved a multi-vehicle defense of a high-valued asset. The high-value asset was deployed to a stationary or at relatively slow-moving trajectory compared to the defense vehicles. In the initial phase shown in Figure 4, eight vehicles are deployed to encircle the high-value asset at a prescribed radius with the HVA at the center. Each vehicle receives periodic position and trajectory updates of its collaborators, and is equipped with a "soft" COLREGS collision avoidance behavior (respecting only Rule 14, and reverting to CPA avoidance in other cases). A separate app is running on each vehicle to reason about the range fore and aft on each vehicle to the nearest collaborator. This informs the requested speed component of the loiter behavior. In the figure, a blue wedge between vehicles indicates equal distance between both the nearest fore and aft collaborator on the radial trajectory encircling the HVA.

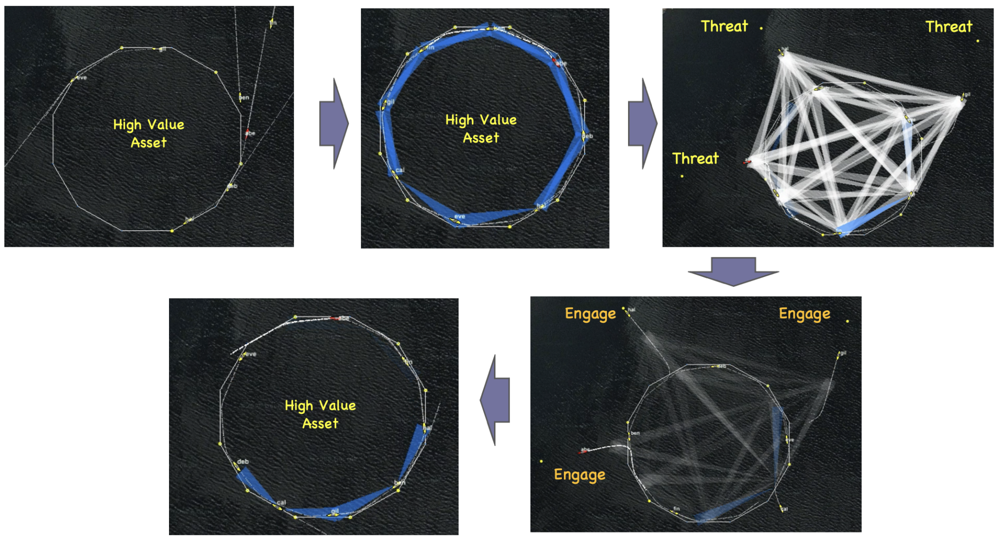


Figure 4: **Swarm Defense of High Value Asset**: Phase 1: Group surrounds and protects a "high value asset". Vehicles are only aware of each other's position, heading and speed Collision avoidance behaviors coordinated with mission behaviors. Phase 2: Members of the group intercept identified threats Vehicles conduct inter-vehicle auction. Selected vehicle assigned. Phase 3: Resume posture.

When a threat or set of threats emerge, the group of vehicles is tasked with choosing a single vehicle to engage each threat. At this point the task manager on each vehicle will spawn a task behavior to participate in an auction associated with each threat. Vehicles are selected and break off from the encircling defensive posture to engage with each threat. During this period the remaining vehicles shift position to retain equal spacing around the HVA. As each engaging vehicle completes its task, it returns to the defensive posture, using the collision avoidance behavior to rejoin the group.

The software modules involved in this mission are `lib_helmtask`, `lib_bhv_waypoint`, `pEncircle`, `pTaskManager`, `uFldTaskMonitor`, in addition to core IvP Helm apps such as the contact manager, collision avoidance behavior, and loiter behavior.

1.6 Example Mission: Convoying

The second example mission involves linear convoying. Such a mission can be described as having two distinct phases. In the first phase, the group of vessels has been given the command to *form* the convoy, which include reaching a consensus on ordering and maneuvering into a convoy formation. In the second phase, the goal is to *maintain* the convoy formation, including holding relative ranges between vessels. The latter phase is handled with a distinct behavior in the Swarm Autonomy Toolbox discussed in Section **??**. The ordering consensus involved in the first phase, is handled using a cascading auction approach discussed here.

The sequence of events for a four vehicle convoy is described in Figure 5. The initial command originates external to the convoy, resulting in an auction to choose a lead vehicle. A task behavior on each vehicle participates in the bidding. On the winning vehicle, the task behavior will (a) spawn a transiting behavior for the lead vehicle to commence, and (b) post a task to the remaining vehicles to choose a follower of the lead vehicle.

On the vehicle winning the number two position, the task behavior will also initiate two events. It will (a) spawn a convoying behavior to follow the lead vehicle, and (b) post a task to the remaining vehicle to choose a follower of the number two vehicle. This continues until the last vehicle is identified and begins following the other vehicles.

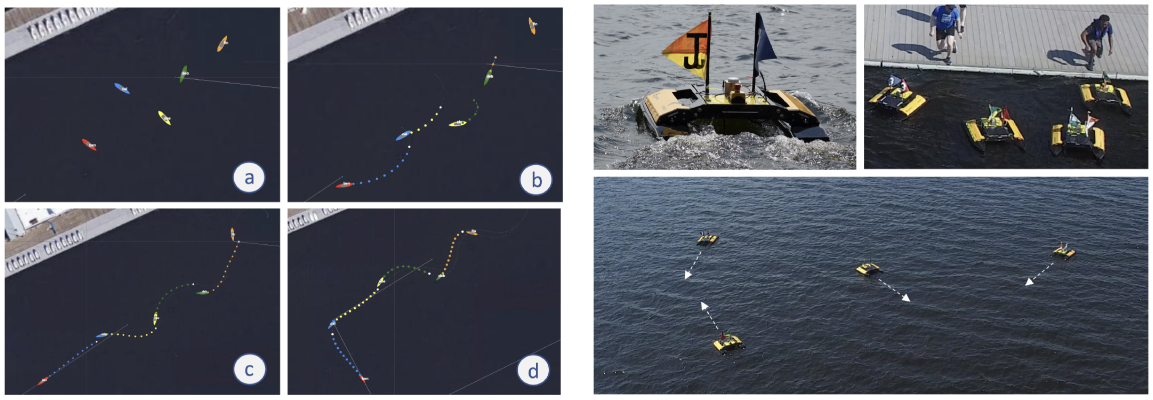


Figure 6: **Convoy Simulation and Field Testing**: Left: (a) Five random starting positions are chosen on each simulation launch, (b) Initial command is given and vehicles self-select their leader and order, as in Figure 5, (c) The vehicles begin to establish their ideal convoy range, (d) The leader has reached the first waypoint and the second vehicle is about to achieve the corner waypoint as well. Right: Four Heron Unmanned Surface Vehicles (USVs) were used for field experiments (top left). Four vehicles were deployed with no assigned roles, or control over starting positions or headings (top right, bottom).

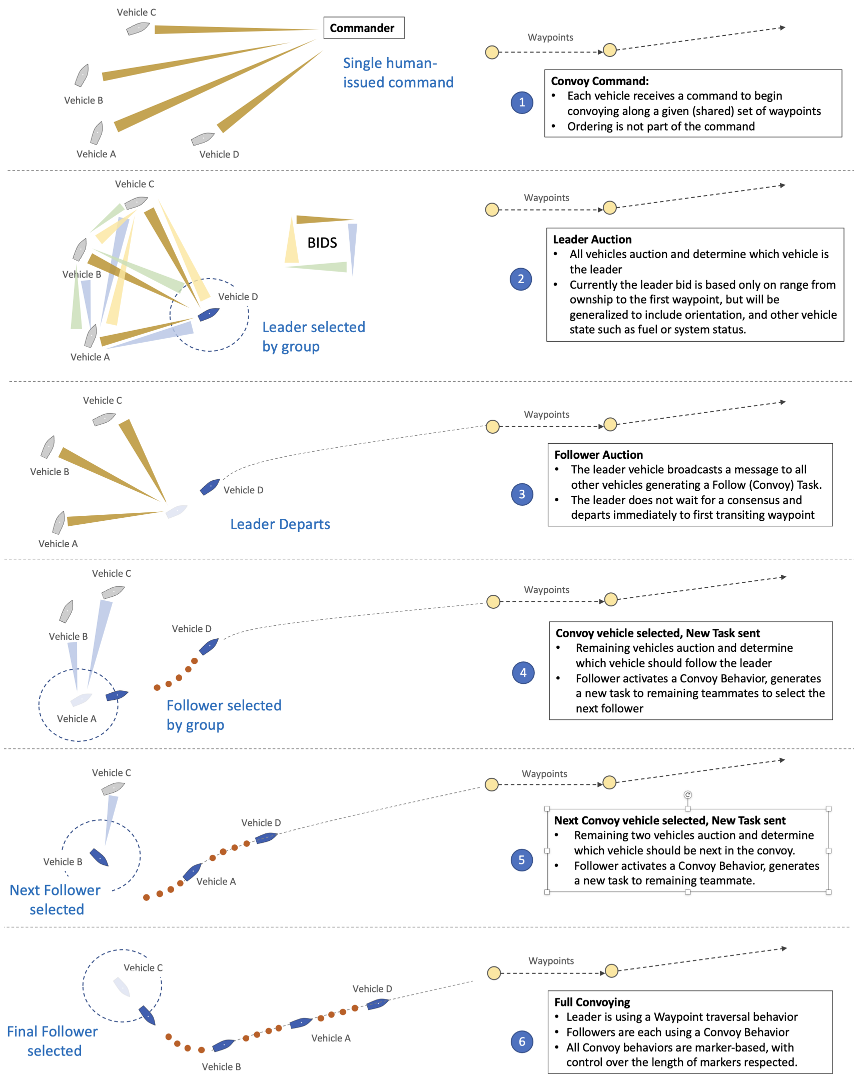

Vehicle C
Commander
Single human-issued command
Vehicle B
Vehicle A
Vehicle D
Waypoints
1
Convoy Command:
• Each vehicle receives a command to begin convoying along a given (shared) set of waypoints
• Ordering is not part of the command
BIDS
Leader selected by group
2
Leader Auction
• All vehicles auction and determine which vehicle is the leader
• Currently the leader bid is based only on range from ownship to the first waypoint, but will be generalized to include orientation, and other vehicle state such as fuel or system status.
Leader Departs
3
Follower Auction
• The leader vehicle broadcasts a message to all other vehicles generating a Follow (Convoy) Task.
• The leader does not wait for a consensus and departs immediately to first transiting waypoint
Follower selected by group
4
Convoy vehicle selected, New Task sent
• Remaining vehicles auction and determine which vehicle should follow the leader
• Follower activates a Convoy Behavior, generates a new task to remaining teammates to select the next follower
Next Follower selected
5
Next Convoy vehicle selected, New Task sent
• Remaining two vehicles auction and determine which vehicle should be next in the convoy.
• Follower activates a Convoy Behavior, generates a new task to remaining teammate.
Final Follower selected
6
Full Convoying
• Leader is using a Waypoint traversal behavior
• Followers are each using a Convoy Behavior
• All Convoy behaviors are marker-based, with control over the length of markers respected.

Figure 5: **Convoy Mustering**: An operator sends a single command to the N stationary vehicles, to traverse a set of waypoints. The group first holds an auction to self-identify a lead vehicle. The lead vehicle then begins a cascading set of tasks to the remaining vehicles to successively hold auctions to determine remaining order. On each selection the proper autonomy behavior is also activated - a waypoint behavior on the lead vehicle and the convoy behavior on the remaining N-1 vehicles.