REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
2/28/2011	Final	3 Nov 2008 - 31 Dec 2010

4. TITLE AND SUBTITLE
Refueling Strategies For a Team of Cooperating AUVs

5a. CONTRACT NUMBER

5b. GRANT NUMBER
N00014-09-1-0198

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Kyle DeMedeiros and Ramprasad Balasubramanian

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Massachusetts Dartmouth
285 Old Westport Rd.
No, Dartmouth, MA 02747

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Office of Naval Research
495 Summer St.
suite 627
Boston, MA 02210

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

13. SUPPLEMENTARY NOTES

14. ABSTRACT
This work presents a study to enhance the Guidance and Control of a vehicle or group of vehicles with a priori and real-time inputs of team status information to manage energy consumption, specifically through refueling, while maintaining mission objectives. In scenarios where 24/7 mission operations are needed, issues arise when vehicles need to enter or leave the operations area to replete their energy supply. In these instances, gaps in coverage can form where the vehicles were stationed, and multiple vehicles may leave at the same time, potentially leaving the entire operations area empty. A novel token passing scheme is used in conjunction with a refueling algorithm which takes in both a priori and real-time data from UUVs (unmanned undersea vehicles) to accurately monitor and manage the rate at which vehicles leave the operations area to replete their energy supply. This algorithm is robust enough to manage the

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Ramprasad Balasubramanian
				37	19b. TELEPHONE NUMBER *(Include area code)* 508 999 8259

Refueling Strategies For a Team of Cooperating AUVs

Kyle David DeMedeiros & Chris Duarte
Naval Undersea Warfare Center
Newport, RI

Dr. Ramprasad Balasubramanian
Computer and Information Science
Dartmouth, MA

January, 2011

Abstract—This work presents a study to enhance the Guidance and Control of a vehicle or group of vehicles with *a priori* and real-time inputs of team status information to manage energy consumption, specifically through refueling, while maintaining mission objectives. In scenarios where 24/7 mission operations are needed, issues arise when vehicles need to enter or leave the operations area to replete their energy supply. In these instances, gaps in coverage can form where the vehicles were stationed, and multiple vehicles may leave at the same time, potentially leaving the entire operations area empty. A novel token passing scheme is used in conjunction with a refueling algorithm which takes in both *a priori* and real-time data from UUVs (unmanned undersea vehicles) to accurately monitor and manage the rate at which vehicles leave the operations area to replete their energy supply. This algorithm is robust enough to manage the refueling of multiple AUVs (Autonomous Undersea Vehicles), in the undersea environment where communications using acoustic modems are noisy and intermittent. The focus of this algorithm is to allow only a specific number of vehicles to refuel at any given segment (a segment being the time it takes for a vehicle to leave the survey area, refuel, and return), and do so under the limitations of intermittent underwater communications while still maintaining accurate information about team and mission state. The intent of this algorithm is to solve a common problem in autonomous team scenarios. A typical refueling scenario involving a group of homogeneous vehicles with the same energy payload and not using a specific algorithm will cause every vehicle to leave at the same time (assuming that their refuel thresholds, the energy level at which they will leave for refueling). This algorithm solves this problem by staggering refueling starting at a high energy level, reducing the negative impact of multiple vehicles leaving at the same time out of necessity. As a result of developing the algorithm with homogeneous AUVs in mind, heterogeneous AUVs can also benefit from this algorithm, as they are highly likely to have different energy discharge rates. In conjunction with this staggering effect, the token passing scheme allows only a specific number of vehicles to leave to refuel, (this is governed by the number of tokens made available to the group). Unless a vehicle's energy level is below a critical threshold, (a level where the vehicle must leave to refuel or risk having insufficient energy to reach the refueling point), the only time a vehicle can leave to refuel is if it holds a token.

I. INTRODUCTION

Unmanned Undersea Vehicle (UUV) team operations are of increasing interest to the US navy. The potential to remove a person or people from harm's way and to efficiently complete mundane tasks is growing in regard to autonomous vehicle technology. There are many tasks suitable for a UUV outlined in [3]. These tasks are prioritized by their interest to the Navy.
- Intelligence, Surveillance, and Reconnaissance (ISR)
- Mine Countermeasures (MCM)
- Anti-Submarine Warfare (ASW)
- Inspection / Identification
- Oceanography

- Communication / Navigation Network Nodes (CN3)
- Payload Delivery
- Information Operations (IO)
- Time Critical Strike (TCS)

These tasks are varying and not only require differing hardware, but potentially different sizes of vehicle. In order to accommodate the varying types of tasks specified for UUVs, four classes of UUV are available.

- Man Portable-25 to 100 pounds;
- Light Weight-500 pounds;
- Heavy Weight-3000 pounds;
- Large-20,000 pounds.

Typically, if missions run only a few hours, a vehicle's energy payload is large enough to complete the mission. When missions become longer, depending on the vehicle used, refueling becomes necessary. In continuous mission scenarios, such as patrolling an area for intrusion, refueling becomes a common and necessary behavior for all autonomous vehicles. A second problem for many UUVs is communication. Acoustic transmissions are used under water as typical RF does not
work without extremely low frequencies or very high power [4]. When transmitting using acoustic waves in a constantly moving medium such as water, the chances for lost or corrupt transmissions are high. The focus of this work is to implement a refueling strategy for Autonomous Undersea Vehicles, (AUVs). The refueling strategy will utilize a token passing scheme which will allow each vehicle to maintain a lock and key on the refuel task, forcing a specific quorum of vehicles to
stay on task at any given time. This token also allows vehicles to confirm messages send from other vehicles, further increasing communications connectivity. The next section will describe some previous work which was involved with, or utilized by this work, followed by details about The Mission Oriented Operating Suite (MOOS), which was the software suite used to create the vehicle behaviors necessary for testing. Section 4 will explain the Refueling algorithm and its token passing scheme. Section 5 will explain the various types of testing done on the refuel algorithm and the token passing scheme. Section 6 will contain the results, conclusions and future work in regard to both the refuel algorithm and the token passing scheme.

II. PREVIOUS WORK

A lot of work has been done in the field of distributed autonomous control. In this work, three main points factor into effective autonomous control:
1. Communications - How reliably teammates can share information with one another. This includes team awareness and consensus among teammates;
2. Distributed Control - How well vehicles can determine what to do on their own with limited state information;
3. Task Management - What the vehicle will do;

A. Communications

Underwater communications rely on the usage of acoustic modems. These modems operate by vibrating a transducer underwater, not by traditional RF signal. This signal can be heard by all vehicles in range capable of picking up the frequency at which the sending vehicle vibrates its transducer. There has been much research on the issue of underwater communications. Much of this research, [5], [6], [7] focuses on the protocol of acoustic modems to maximize communication between vehicles. Some protocols [8], focus on reliable communications while taking into consideration the energy demands of the acoustic modem, while others [9], [10] worry about packet loss and node hopping (sending a packet from source to destination via intermediary nodes). [11], [12], [13], [14], [15] specify the volatile nature of underwater communications, such as limited use of radio frequencies underwater (radio communications only work within the range of 30-300Hz and require large antennae and high transmission power [14]), loss of equipment due to the volatility of the ocean environment and changes in transmission speed of acoustic transmissions. [11] advocates the use of low-cost modems in underwater domains due to oceanic drift, battery replacement, and environmental damage. [14] and [15] note the effect of temperature and salinity on underwater acoustic transmissions. These works highlight the need for distributed control in a team scenario. This work takes into account the volatile nature of acoustic communications in order to utilize team information while still being able to perform the task of refueling in the event of intermittent communication. By enforcing range restrictions and confirmation among vehicles, communication between vehicles can be assured.

B. Distributed Control and Task Management

A central management structure is not feasible in an underwater scenario due to intermittent communications [12], [13]. Each vehicle is not guaranteed to be close to, or have a signal strong enough to always receive information from a central database. [12], [13] point out the infeasibility of a centralized control scheme due to unreliability of underwater communications. These works focus on target tracking and show that a delay in communications between vehicles degrades the confidence of target tracking. Task management allows a group of vehicles to perform specific tasks independently toward the completion of the mission. Due to the volatile nature of underwater communications, many researchers focus on using distributed control to allow vehicles to determine their own tasking, based on intermittent communications between teammates. Task Management is a typical function which needs to be present on all vehicles and needs to be performed continuously in order to effectively complete the mission parameters. By allowing vehicles to manage their own tasks using updates from other teammates, you remove the reliance on a central manager, and thus the constraint a centrally managed underwater network imposes on the mission. Task management utilizing Robust Decentralized Task Assignment (RDTA) was performed in [16]. This Task management utilizes a two-phase approach: Information consensus and Planning. During the Information consensus phase, each vehicle shares information with the other vehicles to sync their state information. During the planning phase, they determine the priority order in which they can perform all tasks. After the priority list is complete for each vehicle, it is transmitted to all other vehicles. each vehicle then simultaneously determines the global task list for each vehicle independently of all other vehicles. The idea is that each vehicle, with the same information about the team and the mission, and running the same algorithm, will each reach the same conclusion. A modification of this algorithm, called Receding Horizon Task Assignment (RHTA) was used in [17]. Instead of each vehicle computing the priority list for every task at once, the prioritization is broken into iterations. Typically, each iteration uses a maximum of three tasks. This iteration effectively reduces the computational complexity. A bid based approach to task management has also been studied as a possible means of decentralization of group task management [18], [19], [20]. This approach can be costly in communications bandwidth and may break down if proper fail-safes are not in place for use in the underwater domain, such as a fuzzy logic to make decisions with less communications. Typically, bid-based approaches use the Contract-Net Protocol (CNP). CNP works as follows:

- A manager sends out a request for bids on a particular task.
- Any node that bids on the task uses its utility as the parameter. This utility is a number depicting how capable the vehicle is in completing the task
- The manager will then look at all bids and determine the best utility, granting the highest utility node the bid.

Utility can be modified by many factors including whether or not the node is currently performing another task. In [18], ground robots perform distributed task allocation using the ASyMTRy-D algorithm, which is based on CNP and shares sensor information between robots. The ASyMTRy algorithm was initially centralized, allowing one vehicle to know all state information about the group. To allow this approach to be more robust, each vehicle runs the decentralized version of the algorithm (ASyMTRy-D), asking for the information it believes it needs from its teammates. Because each vehicle does not know all state information, the optimal solution is not guaranteed. Although the optimal solution isn't guaranteed, robustness is increased as sensor information is shared between vehicles. As such, multiple vehicles can perform the same tasks with different combinations of sensor information, which allows the team to complete tasks in the event that a vehicle becomes inoperable.
In [21], [22], basic CNP is utuluized in the underwater domain. Each vehicle is capable of becoming the CNP manager and all vehicles are able to bid on any task. This utilizes a lot of underwater communication which further illustrates the need for robust under water control. Ulam states that while using their auction based approach, the runtime design requires relatively frequent amounts of communication throughout the mission. In adverse communication environments, mission performance may degrade as robots are unable to receive offers or provide status updates. A token-based approach was used for task management in [1], [23]. In this approach, a token is represented by a value and a vehicle is represented by a variable. A vehicle cannot perform a task unless it contains the token. A vehicle can only perform one task at a time and once a vehicle receives a token, it is locked from receiving any other tokens, so it can complete the task before accepting any others. In the event that there is a task which requires multiple vehicles, in order to allow them to perform tasks while all vehicles are picked to complete the larger task, potential tokens are used. These potential tokens are placeholders for real tokens. Once all of the potential tokens are accepted, the real tokens are then passed out so that the task can begin. In [1], Farinelli adds to this functionality by implementing acknowledgements. With an acknowledgement, the sending vehicle knows that the receiving vehicle has received the token. This removes any uncertainty as to whether the token was received or not. Scerri and Ferinelli's work involves multi-agent systems, that is, systems involving large numbers of computer controlled "bots" that perform specific tasks. This token

passing algorithm decreases the complexity of message passing by an order of magnitude compared to other task management approaches. Fig. 1 illustrates this. Ferinelli adds to this work by implementing it in the multi-robot domain. He utilizes a maximum of 12 robots and implements the token passing using wireless communications, showing the possible uses of this approach in the material world.

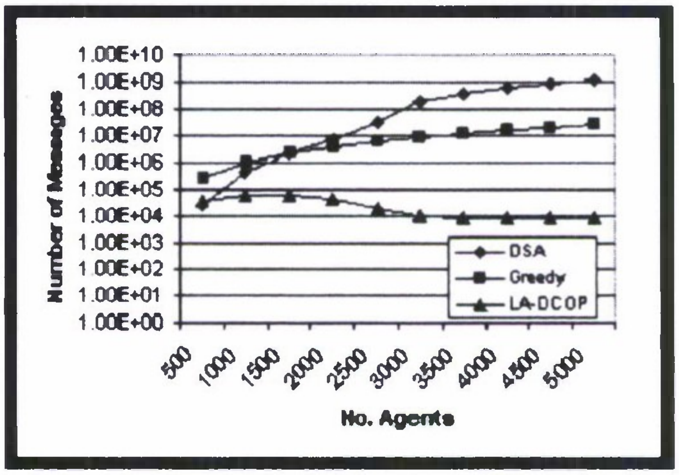

Fig. 1. The number of messages sent versus the number of agents. LADCOPwas the work done in [1]. Note the significant decrease in messagesas a result of the token passing scheme used by Scerri.

C. Contributing Work

Two software packages and a communication protocol were used for creation and testing of this work, the Distributed Control Environment (DiCE), The Mission Oriented Operating suite (MOOS), and the Distributed Interactive Simulation (DIS). MOOS is the principal component where this work was created and utilized, and is described in greater detail in the next section.

1) Distributed Control Environment (DiCE): The Distributed Control Environment (DiCE) [24] is a set of libraries and applications that allows connectivity among processes through shared memory. Within its paradigm, any executable can become a DiCE process. As an added benefit, communication among distributed system components is also facilitated. The goal of DiCE is to facilitate the creation and implementation of robust multi-robot systems. DiCE provides a set of libraries which allow any program to become a DiCE process. In addition, DiCE provides its own syntax, built upon the standard C programming language, which allows for easy translation from DiCE behavior syntax into a standard C program using DiCE Libraries. DiCE was originally intended for use in behavior-based robotic control, but is capable of both reactive and deliberative processes, high level decision making, as well as low-level control. As a result, DiCE can be used as an intermediary between heterogeneous system components. Communication in DiCE is achieved using shared memory segments called ports. These ports allow for any number of DiCE processes to communicate with one another. Furthermore, these ports can be configured with specific attributes such as a queue, latest data, logical AND and OR, and mathematical computation of all *non-read* data in a port. This customization facilitates many behavioral needs.

2) Distributed Interactive Simulation (DIS): An important aspect of any simulation environment is communication. Currently, communications to and from DiCE processes are through shared memory and communications to and from MOOS applications are through the MOOSDB. There are cases where a standard message format is needed. To this end, the Distributed Interactive Simulation (DIS) is used. DIS is not a simulator package. It is a standard for communication. DIS "is a government/industry initiative to define an infrastructure for linking simulations of various types at multiple locations to create realistic, complex, virtual worlds for the simulation of highly interactive activities" [25]. This work uses DIS's Protocol Data Units (PDUs) to pass messages over a DIS "backbone" which is housed on Windows. This backbone is written to by a DIS Publisher, which is a DiCE process, and read by the DIS Reader, another DiCE process. The purpose for including the DIS backbone is to allow for additional software to read information from the simulation. The DIS PDUs "are exchanged between simulation applications and simulation management. The PDUs provide information concerning simulated entity states, the type of entity interactions that take place in a DIS exercise, and data for management and control of a DIS exercise." [25]. There are many PDU types. For the purposes of this work, only an object's physical presence in the system is needed. This representation is called an entity. This entity PDU contains the type of entity, its location, orientation, and its potential appearance to others. For a detailed description of DIS and its standard, please see [25]. In order to view the simulation, a simple graphical simulator was created. This simulator, called Entity Viewer, is another DiCE process which reads entity PDUs. This process gets its input from DIS Reader.

3) Team Knowledge: In order for the refueling behavior to work, team state information is needed. This information, along with other information necessary for other vehicle behaviors, is placed in a status message and broadcast to all vehicles in range. The status message is shown in Table I.

All sections above nav_x of the message is the message's Automated Information System (AIS) report. This is a simple message which holds the name of the vehicle, its speed, its positional information, and its type. This section is written differently for compatibility with third-party software that would read the status message on-board monitoring ships. The vehicle's ID is known to other vehicles based on the vehicle name present in this AIS report. The remaining variables are as follows:

- Type - The brand type of the vehicle, such as a MARV, IVER, etc.
- nav-x - The vehicle's current relative X position
- nav-y - The vehicle's current relative Y position
- Speed - The vehicle's current speed
- Heading - The vehicle's current heading
- Depth - The vehicle's current depth
- Altitude - The vehicle's current altitude
- Power - The vehicle's current energy level, (a percentage)
- Current-plan - Represents the number of vehicles in the group still patrolling the area
- Current-task - The current zone position of the vehicle
- Old-task - The last zone position of the vehicle
- Next-x - The relative x position of the vehicle's next waypoint
- Next-y - The relative y position of the vehicle's next waypoint
- zone0 - Represents the last vehicle who left the patrol area, (uses Vehicle ID)

TABLE I
PROPERTY OF THE NAVAL UNDERSEA WARFARE CENTER. THIS SCENARIO UTILIZES A SIX-VEHICLE GROUP.

Value	MOOS Variable	Type	Range	Precision	# of Bits
_ccl_id		int	[0, 255]	0	8
_id		int	[0, 511]	0	9
Timestamp		int	[0, 131071]	0	17
Node	VEHICLE_NAME	int	[0, 31]	0	5
_dest_id		int	[0, 31]	0	5
_multimessage_flag		int	[0, 1]	0	1
_broadcast_flag		int	[0, 1]	0	1
_unused		int	[0, 3]	0	2
Type	VEHICLE_TYPE	enum	{kayak, asc, auv, ship, buoy, glider, usv, unknown}	0	4
nav_x	NAV_X	float	[-16000, 16000]	0	15
nav_y	NAV_Y	float	[-16000, 16000]	0	15
Speed	NAV_SPEED	float	[-2, 9]	1	7
Heading	NAV_HEADING	float	[0, 360]	1	12
Depth	NAV_DEPTH	float	[-1, 203]	1	11
Altitude	NAV_ALTITUDE	float	[-1, 203]	1	11
power	BATTERY_PERCENT	float	[0,100]	1	10
current_plan	CURRENT_PLAN	int	[0, 6]	0	3
current_task	CURRENT_TASK	int	[0, 6]	0	3
old_task	OLD_TASK	int	[0, 6]	0	3
next_x	SURVEY_NEXT_WPT_X	float	[-16000, 16000]	0	15
next_y	SURVEY_NEXT_WPT_Y	float	[-16000, 16000]	0	15
zone0	ZONE_0	int	[0, 14]	0	4
zone1	ZONE_1	int	[0, 14]	0	4
zone2	ZONE_2	int	[0, 14]	0	4
zone3	ZONE_3	int	[0, 14]	0	4
zone4	ZONE_4	int	[0, 14]	0	4
zone5	ZONE_5	int	[0, 14]	0	4
zone6	ZONE_6	int	[0, 14]	0	4
token_id	TOKEN_ID	int	[0, 6]	0	3
team_id	TOKEN_TEAM_ID	int	[0, 6]	0	3
token_team_1	TOKEN_TEAM_0	int	[-1, 13]	0	4
token_team_2	TOKEN_TEAM_1	int	[-1, 13]	0	4
token_team_3	TOKEN_TEAM_2	int	[-1, 13]	0	4
mission_go	MISSION_GO	enum	{true, false}	0	2
mission_stop	MISSION_STOP	enum	{true, false}	0	2

III. MISSION ORIENTED OPERATING SUITE (MOOS)

A. Core-MOOS

The Mission Oriented Operating suite, (MOOS), was created by Paul Newman at The Massachusetts Institute of Technology (MIT) and is "a set of libraries and applications designed to facilitate research in the mobile robotic domain." [26] MOOS makes available standard applications for mobile robots, but is not limited specifically to that domain. Any application can make use of MOOS and its unique applications. MOOS has a large active community and is frequently updated and maintained. Every MOOS module utilizes a configuration file. If every configuration block is housed in one file, the file is named a moos file. If each configuration block is separate and will be loaded at run-time into a moos file for use, it is called a plug file. Each module's configuration block houses specific information necessary for that module to run correctly such as starting data like initial position information, guard flags, like when to deploy, etc. In this project, three Core-MOOS modules were used, the MOOSDB, pLogger, and pAntler. The MOOSDB is the central hub of all MOOS modules. Any module that needs to communicate with any other module needs to be connected to the MOOSDB. This connection is achieved by subscriptions from and publishing to the MOOSDB through MOOS variables. The MOOSDB

- zone1 - Represents the vehicle present in zone 1, (uses Vehicle ID)
- zone2 - Represents the vehicle present in zone 2, (uses Vehicle ID)
- zone3 - Represents the vehicle present in zone 3, (uses Vehicle ID)
- zone4 - Represents the vehicle present in zone 4, (uses Vehicle ID)
- zone5 - Represents the vehicle present in zone 5, (uses Vehicle ID)
- zone6 - Represents the vehicle present in zone 6, (uses Vehicle ID)
- token-id - The number representing the ID of the owner of the token
- team-id - The number representing the team the vehicle is on
- token-team-1 - The number representing the owner of the token, (based on token ID) for sub-team 1
- token-team-2 - The number representing the owner of the token, (based on token ID) for sub-team 2
- token-team-3 - The number representing the owner of the token, (based on token ID) for sub-team 3
- mission-go - Determines if the mission is running
- mission-stop - Used to stop the mission

Zone0 to Zone6 are used to determine how the team perceives where everyone else is. These variables hold the zone positions perceived by each vehicle. Every vehicle knows the start zone for each vehicle. As vehicles enter and leave, this information is updated based on incoming messages. Zones are positional rectangles in the survey area. One vehicle is present in each zone. As vehicles leave to refuel, zones merge, and as vehicles return from refueling, zones split. This work involves a wedge-shaped survey area, broken into at most 6 zones, at least 3. This is explained in Section5.

The token is the mechanism which allows a vehicle to leave to refuel. The token-id is an ID independent of the vehicle's standard, (acoustic modem) ID. Every vehicle is given a token-id based on its team-id. For example, if there were two tokens, a 6-vehicle group would be split into 2 sub-teams, represented by token-team-1 and token-team-2 each team would have three vehicles. Each of these vehicles would be given a token-id between 0 and 2. The combination of team-id and token-id allows the token to be passed from vehicle to vehicle within the sub-team. This work focuses on a 6-vehicle group and this scenario allows a maximum of three sub-teams.

monitors all MOOS variables and each module subscribed to the MOOSDB will be notified when an update to their respective subscriptions occurs. This allows all modules to be connected to each other through a star-like topology. All information passed is accessible by any module, assuming that the module is subscribed to the appropriate MOOS variable. Furthermore, every application has the ability to publish to the MOOSDB. Both publishing and subscribing is facilitated through a mail system. MOOS messages house the variable and the time it was published. This allows an application to look for a specific message stored in the MOOSDB, or, in a simpler case, a module may only need the latest update and therefore only the latest time-stamp will be sent to the module. pLogger is used to read information passed through the MOOSDB. Any variable sent through the MOOSDB is accessible. It is also possible to change the rate at which pLogger records the information, to a maximum of the speed the MOOSDB publishes the variable. For example, if the MOOSDB publishes a variable A 20 times per second (20Hz). pLogger is capable of recording this information at a maximum of 20Hz. If the speed is increased to any faster, cycles of no work will occur. pLogger can also determine which variables it will and won't log. for some tests, you may only need specific variables, but for others, you may need everything. When pLogger records everything in the MOOSDB, this is called wildcard logging. Every variable housed in the MOOSDB will be logged at the rate specified in pLogger's configuration file. pAntler is a module used to run any MOOS module. This module allows a series of other modules to be run, specifies and order, and specifies a time spacing between module launches. This removes the need for the operator to manually run each module or to write a script to run each module. When pAntler is used, a moos file containing all the configuration blocks of each module to run is needed as pAntler runs each module with its configuration information.

B. MOOS-IvP Built on top of MOOS is MOOS-IvP.

MOOS-IvP was created by Mike Benjamin in 2004 "for autonomous control of unmanned surface craft and later underwater vehicles." [2] The focus of these applications and libraries is to solve
multi-objective problems and facilitate command fusion of multiple behaviors in MOOS. Basic MOOS has applications, interfaces, behaviors, and utilities. MOOS-IvP extends MOOS's original module set and is typically packaged together with Core-MOOS. In this project, pHelmIvP and pNodeReporter were utilized. pNodeReporter is used to collect positional information including relative x position, relative y position, heading, speed, and depth, packages it into one variable, and publishes it to the 5 MOOSDB for other applications to use. This is a form of AIS report, which is a typical information format for large marine vehicles [2]. pHelmIvP is the main controller for the vehicle. This utilizes MOOS behaviors and command fusion to determine how the vehicle will move. It bridges the high-level command with the low-level control [27]. pHelmIvP utilizes behavior based control and a concept called interval programming, (IvP). See Fig. 2 for a graphical overview of pHelmIvP.

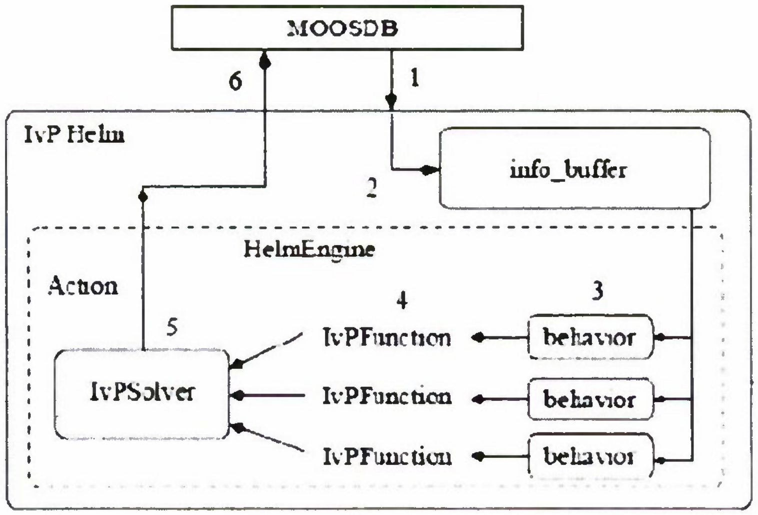

Fig. 2. Components of the IvP Helm and its iteration flow as seen in [2] (1) mail is read from the MOOSDB, (2) mail information is parsed and stored in a local buffer to be available to the behaviors, (3) the conditions required for behavior activity are evaluated for each behavior, (4) active behaviors produce an objective function if applicable, (5) the objective functions are resolved to produce an action which is (6) published to the MOOSDB for other MOOS processes handling lower-level vehicle control to consume.

1) Behavior-Based Control: pHelmIvP subscribes to and publishes MOOS variables like any other process. The difference here is that all veriables subscribed/published to by pHelmIvP are requested by behaviors that pHelmIvP runs. Behaviors consist of processes which have direct influence on actuator control of the vehicle such as collision avoidance or waypoint navigation. Each bevavior outputs an IvP function which effects how the vehicle will move. these IvP functions are sent to pHelmIvP's IvP solver which utilizes a branch and bound search over the combination space of all of the functions. Pruning techniques are also used to speed up the search, [27].
2) Interval Programming (IvP): Interval Programming is a method of representing and solving multi-objective optimization problems. It utilizes piecewise linear functions to estimate the behavior's utility function. This allows the IvP solver to be independent of function type as all functions are mapped to piecewise linear approximations, making the solver appropriate for virtually any function. The burden is then placed on the behavior programmer to pick the appropriate function for the behavior and the level of detail that the linear representation will have [27].

C. MOOS-IvP-Local

MOOS-IvP-Local is a collection of MOOS modules made by various authors housed at MIT. MOOS-IvP-Local houses a large number of MOOS modules. For the purposes of this project, one main module was used, pGeneralCodec and pAcommsHandler. These two applications allow MOOS to talk using an acoustic modem, allowing communications between teammates. pGeneralCodec is an encoding and decoding module. It allows data from the MOOSDB to be encoded into an acoustic message. When a message is received, pGeneralCodec decodes the message and places the information into the MOOSDB. XML configuration files are used to determine which variables are read from the MOOSDB for message encoding and which variables are written to for message decoding. This removes the restraint of a specific interface when using pGeneralCodec [28]. pGeneralCodec creates a Woods Hole Oceanographic Institute (WHOI) modem hexadecimal string message. pAcommsHandler is used to manage queuing of massages. It allows the user to determine the priority of message types, whether or not messages in the queue should be used or discarded, (queue pruning), and what type of queue is used, (FIFO, LIFO). pAcommsHandler is also capable of the same functionality as pGeneralCodec, as it contains access to WHOI data structures and contains an encoder/decoder [28]. For this work, pAcommsHandler is not used as an encoder/decoder.

D. MOOS-IvP-Jeul

MOOS-IvP-Jeul is a set of MOOS applications and utilities created by various authors at NUWC. These applications allow for reconfiguration of vehicle formation and tasking, contains the interface to the vehicle's control from the physical (or modeled) world, determines communications timing, and facilitates common state information among teammates. All software created for this project is housed in MOOS-IvP-Jeul. This project utilized pSurvey, iMarv, iOceanServerComms, TDMA, VehicleMap, retask, and pRefuel. pRefuel is explained in detail in the next section.

1) pSurvey: pSurvey is used to modify the waypoint behavior housed in pHelmIvP. This module allows a new set of waypoints, order of arrival, speed, and depth to be sent to the waypoint behavior when retask notifies it of a task change. The pattern used for patrol is a vertical bowtie, 200m squared, housed in each zone. retask transmits these points to pSurvey and pSurvey determines goal radii, (a ring around the waypoint which determines a successful reach of the waypoint), the speed, and depth.

2) iMarv and iOceanServerComms: iMarv and iOceanServerComms are used as the interfaces from the world to the vehicle controller. They receive dynamics information from the world models and update the MOOSDB with this information. Information such as positional information, (latitude, longitude, depth), as well as energy information, is received from the world models by iMarv and iOceanServerComms. They act as the bridge between what is happening and what the vehicle wants to do. The main difference between these two modules is the connection to the controller. iMarv utilizes ethernet and iOceanserver utilizes serial. The vehicles made available are Ocean Server IVER vehicles, which use a serial connection. as a result, any in-water testing uses iOceanServerComms and any simulation uses iMarv.

3) TDMA: TDMA controls the communication scheme of the vehicles. TDMA stands for Time-Division, Multiple-Access. It segregates a time cycle into discrete time intervals, one for each vehicle, in which they are allowed to communicate. During this period, TDMA signals GeneralCodec to create the message and this allows pAcommsHandler begin transmitting. Parameters inside the TDMA configuration block allow it to be customized based on number of UUVs, data rate, message size, and the overall length of the TDMA cycle. This allows the scheme to be reusable based on the mission.

4) VehicleMap: VehicleMap is used to store and compare team state information between vehicles. There are seven bins for positional data, one for each vehicle and one for the last known vehicle that left the patrol area. The positional data consists of the last known zone a vehicle was in. This data is transmitted in a status message using the seven zone variables mentioned in the last chapter. each time a message is received, VehicleMap compares the received bins with its internal representation. If a message is received from a vehicle depicting

that vehicle in a different position than the receiving vehicle's internal representation, the receiving vehicle updates its positional information. If a vehicle needs to leave the patrol area, it places itself in zone 0 and transmits the status message. The receiving vehicle will then compare this message to its internal representation, wipe the leaving vehicle from its old zone, and will transmit the updated information in the status message.

	Veh1	Veh2	Veh1	Veh2	Veh1	Veh2
Refuel	1	0	1	1	0	0
Zone1	1	1	1	2	0	2
Zone2	2	2	2	3	0	3
Zone3	3	3	3	4	0	4
Zone4	4	4	4	5	0	5
Zone5	5	5	5	6	0	6
Zone6	6	6	6	0	0	0
	t1		t2		t3	

Fig. 3. Vehicle Map. At t1 veh1 requests to leave by placing its ID into the refuel position. At t2, veh2 acknowledges by removing veh1 from its zone. At t3 veh1 resets its vehicle map to all 0's indicating it is refueling. Veh2 purges veh1 from the vehicle map.

Every zone must remain connected to its neighbors, therefore, if a vehicle leaves a zone, the highest numbered zone left in the formation is merged with all other zones in its tier and all vehicles reconfigure to maintain the patrol area's integrity. For example, if a vehicle in zone 6 leaves, this zone needs to be merged evenly into all zones present in the same row. The vehicles populating these zones will then reconfigure to accommodate the newly enlarged zones. The opposite occurs when a vehicle returns to the patrol area. The returning vehicle enters the next available zone, and all vehicles reconfigure to allow the zone newly created a place in the patrol area. For example, a vehicle is returning to zone 6. The enlarged zones present in zone 6's row will shrink to accommodate the returning zone and each vehicle in that row will shift.

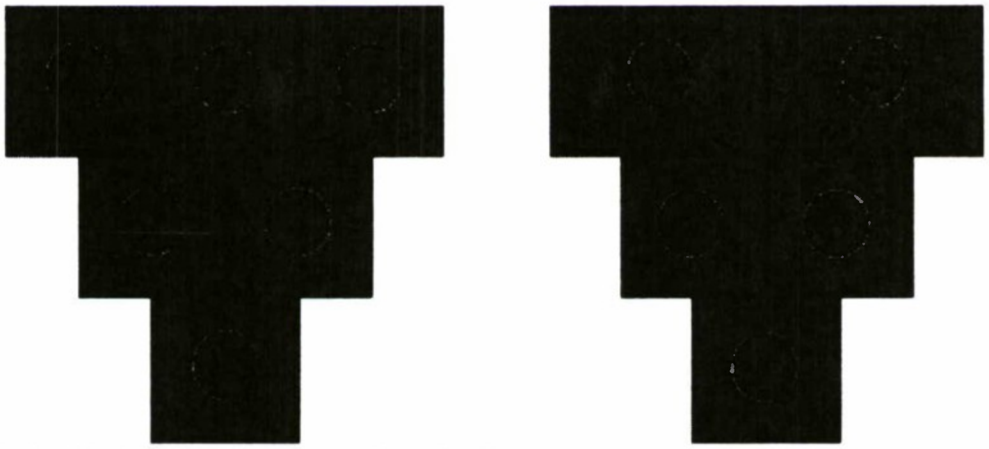

Fig. 4. A 6-vehicle to 5-vehicle reconfiguration. The circled numbers indicate vehicles while the uncircled numbers indicate zones. Notice how vehicle 4 is removed indicating a refuel event from the right image. Vehicles 5 and 6 enlarge their zones to compensate for the gap. The zone numbering also changes.

5) retask: retask is the task reconfigurer. This module monitors VehicleMap to determine the positions of all vehicles, if any vehicles have left, and if any vehicles have joined the group. It also monitors pRefuel to determine if a vehicle needs to leave the patrol area to replenish its

energy supply. If there are any changes to the group that effect the vehicle running retask (i.e. if the vehicle in the zone next to it left, if it has to go refuel, if it is coming back from refuel etc.), retask changes the current task of the vehicle, reconfiguring it with new parameters (zone changing and waypoint reconfigurations).

IV. THE REFUEL ALGORITHM AND THE TOKEN PASSING SCHEME

A key, or token passing is commonly used in distributed computing [29], [30], [1], [23]. In this paradigm, when a critical section of code needs to be accessed, a key is required. Only the holder of the key is allowed access into the critical section. Since only one process can possess the key at any time, the risk of another process reading incompletely written data is eliminated. In an analogous situation, unmanned underwater vehicles (UUVs) also need to share a common resource, in this instance, energy management using refueling. While the mission can continue with a low number of vehicles absent, the simultaneous, albeit temporary, loss of more vehicles imperils mission success, (i.e. vehicle coverage of the survey area). The key scheme applies equally well to this situation. In our case, the key is a token. It will be passed from vehicle to vehicle and allows a vehicle to leave the survey area to refuel, (barring specific circumstances discussed in later paragraphs). The token is passed in a ring-like manner, going from first vehicle to last and back to first, the last pass utilizing a relay vehicle. The mission this algorithm was designed for involves a series of vehicles, at most 6, separated into tiers and zones, surveying an area as discussed in section 2. A tier is a row and a zone is a column. Each vehicle will communicate via broadcast acoustic message using a TDMA scheme. TDMA places each vehicle into a communications interval. The vehicle can only communicate during this period. For the remaining periods, its communication process waits to receive messages from teammates. A TDMA cycle consists of the time it takes from the start transmission of the first vehicle to the fourth start transmission of that same vehicle, i.e. each vehicle transmits 4 times during a TDMA cycle. The need for this restriction arises out of the equipment utilized in the experimentation. An acoustic modem is the communications device modeled. Each vehicle utilizes the same acoustic frequency and therefore can not transmit simultaneously.

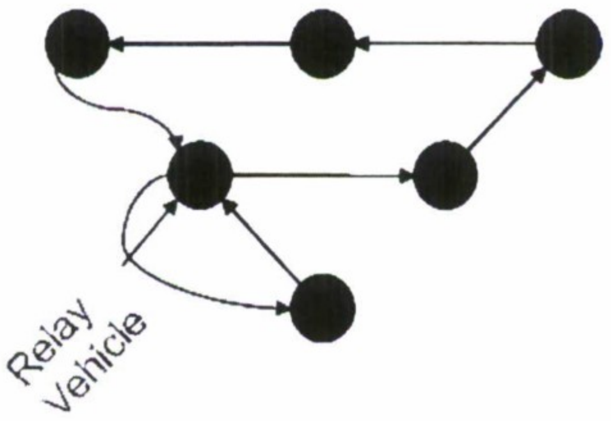

Fig. 5. Initial Order of vehicles and the direction in which the token is passed.

A. The Refuel Algorithm

The design of the refueling algorithm is based on the number of vehicles versus the number of tokens. The algorithm is designed to work with a mission scenario of 6 vehicles or less, with the possibility of at most 3 tokens in use. This constraint is due in part to the size of the message capable of being transmitted through the water (a 32 byte message, using a Woods Hole Oceanographic Institute acoustic modem). To remove gaps in the group formation, when a vehicle leaves to refuel the team reconfigures itself to fill the gap. The more vehicles that leave, the tighter the formation, (zones merge and tiers are dropped). Fig. 11 illustrates this. This reconfiguration is part of another process which monitors when vehicles need to leave or join the group. The refuel algorithm will be used to determine the appropriate time when a vehicle should leave to refuel when collaborating in a group mission. Variables involving a vehicle refuel are as follows:

- Number of refuel tokens for the group;
- The vehicles own Energy Level;
- Energy levels, (or the perceived energy levels) of the rest of the sub-team;
- Communications connectivity, (how often a vehicle hears from another vehicle);
- Each vehicle's distance to the refuel point.

To avoid the possibility of multiple vehicles overwriting the same token number when multiple tokens are available, the set of vehicles will be split into sub-teams based on the number of tokens available to the mission; at least one team, at most three teams. Teams are made based on their location in the survey area. Vehicles in close proximity are assigned to the same sub-team.

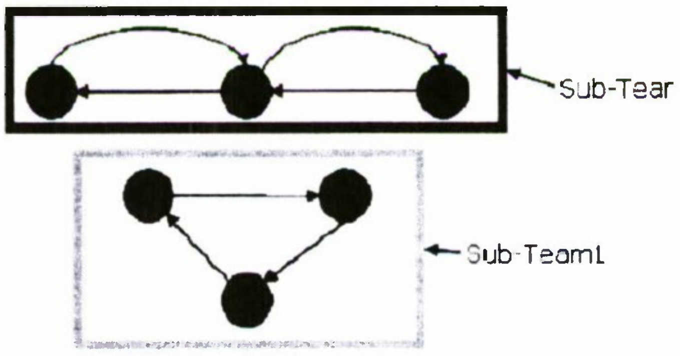

Fig. 6. Initial order of a two-token scenario. Teams are broken down by tier and zone ID. Teams are fixed throughout the mission.

To determine whether or not a vehicle will refuel, it must be aware of its teammates energy levels, speed, rate of consumption, and distance to the refuel point. Using this data, the vehicle can then make an estimate of its own future energy level. There is an energy threshold, denoted as batChgdThresh used to determine the full charge of a vehicles battery. This is set in a configuration file. There is also a threshold, denoted as inCompThresh, used to determine if a teammate vehicle, (a vehicle whose token is shared with the computing vehicle) should be considered in the computation of the future energy level. This value is set in the configuration file as well. If the actual energy level is below the critical threshold, denoted as critThresh, the vehicle leaves regardless of whether or not it has the token. This critThresh is the minimum energy level required to reach the refuel point from where the vehicle currently is. This value is multiplied by 2 so as to retain a conservative estimate in case of obstacles (high current, items blocking a path, etc.). If the computing vehicles future energy level is below a normal (non-critical) refueling threshold, denoted as refThresh, the vehicle leaves to refuel, assuming it has the token. The refThresh has a base value set in the configuration file. This base value is the highest value allowed for the vehicle to leave to refuel.

As more vehicles are considered in the algorithm, this value is lowered. By allowing a vehicle to leave to refuel at a higher energy level, you reduce the possibility of a vehicle being forced to leave as a result of a critical refuel. In order to make the refuel decision, several pieces of information must be accessed and maintained during group communication. The stored and maintained information includes both the calculating vehicles information, the sub-teams information, and some volatile token numbers about other sub-teams, such as which vehicle has the token. (This token information is considered volatile as it is only considered valid by the receiving vehicle until the receiving vehicle transmits it.) A vehicles distance from the refuel point must be continuously calculated as part of the refuel decision. To determine the distance of a vehicle to the refuel point, either a vehicles actual position (if known), or the center of the survey area is used. The distance will then be calculated using basic vector mathematics:

$$Distance = \sqrt{(x1-x2)^2} + (y1-y2)^2 + (z1-z2)^2 \quad \text{(IV.1)}$$

where (x1, y1, z1) is either the known vehicle position or the center of the survey area and (x2, y2, z2) is the refuel point. Using the distance to the refuel point, designated as dist, the following time calculations are made:

$$timeToRf = \frac{dist}{speed} \qquad \text{(IV.2)}$$

where timeToRf is the time to the refuel point and speed is the current speed of the vehicle, or an average speed of the vehicle, if recent data is not available.

$$totTime = \textstyle\sum_0^{n-1} vehicle[i]\!:timeToRf + vehicle[i]\!:rfTime + (TDMA\_j) \qquad \text{(IV.3)}$$

where totTime is the total estimated time before the calculating vehicle receives the token again, n represents the number of vehicles considered in the algorithm, rfTime is the time it takes a vehicle to recharge its battery, TDMA is the time in one TDMA cycle, and j is the number of vehicles not considered in the algorithm. In calculating the total estimated time to regain the token, denoted by tot time, delays in hearing from a predecessor vehicle, (i.e., the vehicle that will be passing the token to the current vehicle), must be taken into account. As the formation of the vehicles is not an actual ring, a relay vehicle is needed to pass the token. Fig. 5 illustrates this.

This relay takes one additional TDMA cycle:

$$totTime = \textstyle\sum_0^{n-1} vehicle[i]\!:timeToRf + vehicle[i]\!:rfTime + (TDMA\_j) \qquad \text{(IV.4)}$$

where j must be at least 1.

$$Vehicle[i]\!:energy = vehicle[i]\!:energy((vehicle[i]\!:RoC \times elapsedTime) \times vehicle[i]\!:alpha) \quad \text{(IV.5)}$$

where elapsedTime is the time since the last computation and RoC is the vehicle rate of consumption (in %/s). In our network, each vehicle must have knowledge of the fuel status of every other vehicle. Because communication may be intermittent, each vehicle estimates the remaining fuel of the other vehicles using the above equation until new data is available. That includes a safety consumption factor _ which increases in a linear fashion. During a period of lost communication, _ increases from 1.0 to 2.0.

$$vehicle[i]:alpha = vehicle[i]:alpha + (alphaIncreaseRate\_ \, x \, elapsedTime) \qquad \text{(IV.6)}$$

where alphaIncreaseRate is the amount α is increased by each second (specified in the configuration file). The total α, denoted by totalAlpha, is the average of each vehicle's individual α's that take part in determining each vehicles assumed energy level. Both α and totalAlpha are at least 1, indicating that the total energy consumed if the token is passed is greater if the communications to a particular vehicle are bad. In order for _ to increase the estimated future energy level of the AUV running the algorithm, it needs to be at least 1 (which indicates a message was just heard) or higher (which will degrade the confidence of the vehicle), causing it to assume more energy will be consumed before it gets the token back.

$$totalAlpha = \frac{\sum_0^n vehicle[i].alpha}{n} \qquad \text{(IV.7)}$$

where n represents the number of vehicles used in the computation of futureEnergy

$$futureEnergy = energyLevel((RoC\_totalTime)\_totalAlpha) \qquad \text{(IV.8)}$$

Where *futureEnergy* is the next estimated energy level based on the next token receipt.
In the computation of the future energy level, the rate of consumption is multiplied by total time to get the energy the vehicle would use up waiting for the token to arrive. The totalAlpha is multiplied by this assumed energy usage. This implements the uncertainty factor due to teammates not in communications range. For this experimentation, the rate of consumption (RoC) is a linear representation of the percentage of energy used per second based on an average speed. Models of Ocean Server IVER AUVs were used with an average speed of 1.25 m/s and an energy payload of 600WHr, lasting 8Hrs. By setting 8 hours to be 100 percent, approximately 0.00347 percent of the vehicles payload will be consumed per second while moving at a speed of 1.25 m/s. This is a base value stored in the configuration file. As the speed of the vehicle changes, the rate of consumption changes, the faster the vehicle moves, the more energy is used. Furthermore, assistance or hindrance by ocean dynamics such as current will modify the energy consumption accordingly. As such, the rate of consumption can be fed into the refuel algorithm to mirror the effects of the ocean dynamics. The focus is on improving the incoming data for a more efficient reactive response to energy consumption. The ocean current information is read from actual in-water tests run with the IVER AUVs. This data is parsed by a processes called iDVLSim, created by Saban Singh of The University of Massachusetts, Dartmouth, and parsed into variables. These variables are read by a model of a Doppler Velocity Log (DVL) interface, which is a form of current profiler using sonar technology which monitors vehicle speed as well as water current speed.

$$relativeHeading = heading \,\square\, waterCurrentDirection \qquad \text{(IV.9)}$$

where heading is the vehicle's heading with respect to true north, waterCurrentDirection is the direction of the water with respect to true north, and relativeHeading is the direction of the water with respect to the vehicle's forward motion By fixing the vehicle's heading as the x-axis of the vehicle, the direction in which the current is moving can be with respect to the x-axis, allowing for the factoring out the water's separate directional vectors.

$$waterXSpeed = cos(relativeHeading) \_ waterCurrentSpeed \qquad \text{(IV.10)}$$

where waterXSpeed is the direction the water is moving with respect to the vehicle's forward motion. A negative value indicates that water is moving against the vehicle's forward motion.

B. The Token Passing Scheme

The token passing scheme determines how the token will be passed. In this work, a token ring is used where the token is passed from vehicle to vehicle starting from the smallest vehicle ID and moving to the highest vehicle ID. The last vehicle will than pass the token to the first via a relay vehicle and the cycle will begin again. The need for the usage of vehicle ID's as well as relay vehicles becomes apparent when you take into consideration reconfiguration. When a vehicle with a token leaves to refuel, it is not guaranteed to return to the same position it left, and therefore passing to the next zone is not necessarily the next vehicle in line to use the token.

TABLE II
TOKEN NUMBERS AND THEIR MEANING.

# of sub-teams	# of vehicles in sub-team	token owner #	refueling vehicle #	sentinel value	unused values
1	6	0-5	6-11	12	–
2	3	0-2	6-8	12	3-5, 9-11
3	2	0-1	6-7	12	2-5, 8-11

There are three separate numbers denoting the token values for each sub-team. These numbers have multiple purposes depending on the actual number stored in the variable. Tokens are to be passed to the next recipient via a status message. The values allow for all of the vehicles to share information about token status, but the values scale depending on the number of teams present. See Table II for a breakdown of values for the token numbers. Refuel numbers are used to inform the team that a vehicle is leaving to refuel. When a vehicle receives the token and needs to refuel, it adds 6 to the token number and this is its refuel number. For example, vehicle 6 (Veh6) is represented by token number 5. It will add 6 to its number to get 11, its refuel number. This number will constantly be sent during its TDMA cycle when it is refueling to inform the team that it is refueling. These refuel numbers are not stored by the vehicles, they are just to inform each vehicle that a teammate is refueling. The sentinel value is used to allow vehicles to ignore the token number when there is no new information. Without this sentinel value, tokens can be missed or duplicated. For example, suppose Veh1 passed the token to Veh2, and Veh2 passed the token to Veh3, but Veh3 did not hear that Veh2 had passed the token to it. Veh3 would then look in memory and see that Veh2, according to it, has the token. This old data would then be sent to the team, making Veh3 miss out on the token.

In a second example, Veh1 passes the token to Veh2, Veh3 does not hear this and on its TDMA transmission time transmits that Veh1 still has the token. Veh1 will now have a token and Veh2 knows it already has a token and disregards the new message as false. Two tokens are now created for one team. This occurrence needs to be reduced as much as possible so that only the allowed number of vehicles can leave to refuel. The token number is stored internally and updated based on the current number in status messages received from teammates. If the sentinel value is seen, it is ignored and the internal representation of the token number stays the same. To guard against the possibility of losing tokens (multiple vehicles each think the other has the token, or everyone now sees sentinel values constantly rather than an updated token number), the vehicle who owns the token (the vehicle that last received the token destined for itself), continually passes the token to the next recipient until it receives an acknowledgement. This confirmation is not a vehicle-to-vehicle confirmation, simply a confirmation from a vehicle that heard that the sending vehicle passed the token. For example, if three vehicles are in range of one another, Veh1, Veh2 and Veh3. Veh1 passes the token to Veh2 and both Veh2 and Veh3 hear it. If messaging is in order, Veh2 will send a status message next with the recently received token number in it. This acts as a confirmation for Veh1. In a second scenario, suppose Veh1 passes the token to Veh2, but only Veh3 hears it. Now suppose Veh2 is in range of Veh3, when Veh3 sends out the status message stating that Veh2 has the token, Veh2 will get the token by proxy, making V3 act as a relay vehicle.
A vehicle becomes a relay vehicle when three requirements are met

- The vehicle is not the recipient of the token;
- The token number received is not what is currently stored in the vehicle's memory;
- The vehicle is not the sender of the token.

Relay vehicles do not enforce acknowledgement due to confusion among teammates. See Fig. 7 for a representation of confirmation confusion.

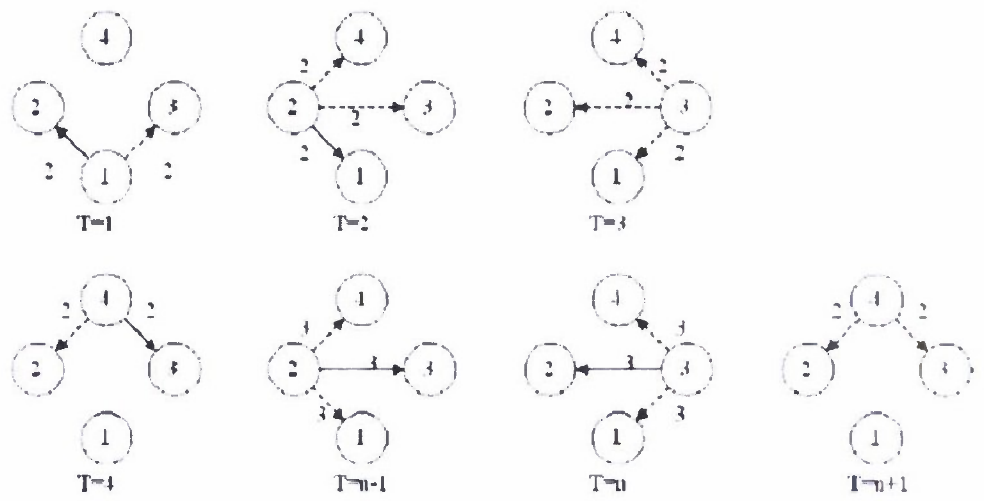

Fig. 7. An example of relay vehicles being forced to wait for an acknowledgement. The solid lines represent the intended recipient of the message. The dashed lines represent all other recipients. In cases where no solid lines are present, the message was intended for any vehicle. Note the potential for group splintering based on token number.

In Fig. 7, four vehicles are on the field Veh1, Veh2, Veh3 and Veh4. Veh1, Veh2, and Veh3 are in range of one another, but Veh4 is only in range of Veh3. At t=1, Veh1 will pass the token to Veh2, Veh3 hears this message. At t=2, Veh2 will acknowledge the token receipt, Veh3 and Veh4 also hear this message. At t=3, Veh3 broadcasts out its known token value (2), requiring an acknowledgement. At t=4, Veh4 acknowledges the message, Veh2 also hears this. at some time t=n-1, Veh2 passes the token to Veh3, Veh1 and Veh4 hear this as well. Because Veh4 never received an acknowledgement, it discards this message. At t=n, Veh3 acknowledges the message, Veh1 and Veh4 hear this. Veh4 ignores the message. At t=n+1 V4 broadcasts out the previous token number, Veh2 and Veh3 hear this. At this point, Veh1, Veh2 and Veh3 know the token is 3, but Veh4 believes the token is at 2. Due to this occurrence, this work does not utilize forced acknowledgements for relay messages. As mentioned earlier, there are cases where the vehicle can enter a critical refuel scenario. In these cases, the vehicles energy level is at a point where it can no longer wait for a token to be passed to it. The energy left is enough to make it to the refuel point (with an additional buffer in case of unforeseen circumstances like strong current, obstacles, etc.). This critical refuel scenario can cause potential problems as no token is required to refuel and there is no restriction on the number of vehicles leaving due to a critical energy level. This option is needed, however, as the vehicle could potentially be lost in the water otherwise. To decrease the number of vehicles reaching the critical level, the above mentioned method of setting the safe refuel threshold (refThresh) high and lowering it as more vehicles are added to the refueling algorithm for consideration acts to stagger the vehicles leaving to refuel.

In those cases where there is more than one team, the need for non-teammates to pass information may still be necessary. There must be safeguards in these scenarios though, as there is more of a chance of intra-team confusion from inter-team communication than from intra-team communication. This is due to non-teammates managing only minimal data about vehicles not in their team. To reduce the amount of old data sent due to inter-team communication, a nonteammate is only allowed to send data about another team that it just received. If the data received is a sentinel value or old data matching the internal representation of the other teams token number, the sending value is a sentinel value. The issues explained above are potential pitfalls in this token approach. They cannot be avoided completely due to intermittent communications, interference, corruption of data, distance between vehicles, and the method in which the token is passed from vehicle to vehicle (in a ring-like manner). It will be shown in Section 6 that other scenarios arose out of experimentation and the refuel algorithm was modified to compensate.

V. EXPERIMENTATION

A. Connectivity and The Basic Refueling Behavior

The experimentation shown below involves the same configuration of vehicles as seen in Fig. 8. Experiments were run to evaluate the effectiveness of communications and the interconnectivity of the team of AUVs as well as to determine both the effectiveness of the refueling application and the emergent behavior among a team of AUVs. these experiments did not use a variable Rate of Consumption. These experiments included multi-token experiments as well as single token experiments, all on a 6 vehicle configuration. 6 summaries of the experiments are shown below:

- Scenario 1: Full battery capacity, 1 token, no refueling
- Scenario 2: Reduced battery capacity (.5-1hr.), 1 token, simulated refueling, critical refuel enabled
- Scenario 3: Reduced battery capacity (.5-1 hr.), 2 tokens simulated refueling, critical refuel enabled
- Scenario 4: Reduced battery capacity (2hr.), 1 token, simulated refueling, critical refueling disabled
- Scenario 5: Reduced battery capacity (2hr.), 2 tokens, simulated refueling, critical refuel disabled
- Scenario 6: Full battery capacity, 1 token, simulated refueling, critical refueling disabled

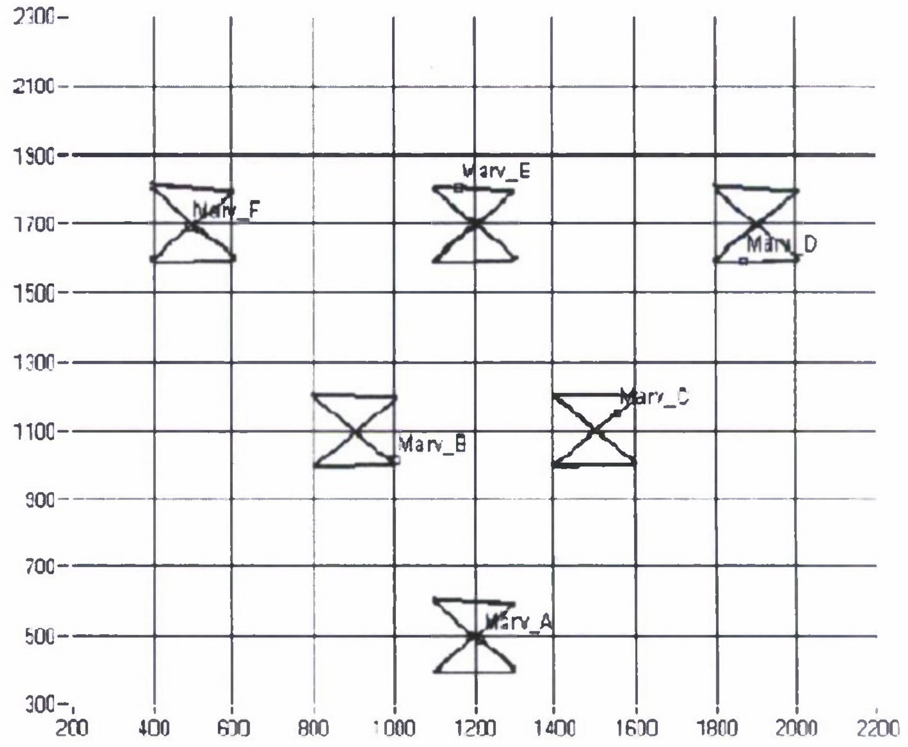

Fig. 8. Vehicle Configuration. The horizontal and vertical axis are in meters(m).

The first scenario is a benchmark to determine connectivity between vehicles. The first two reduced battery capacity scenarios were done to decrease simulation time, (based on speed and energy consumption, the vehicle would need to discharge for 8 hours (normal battery capacity) before hitting 0 energy), as well as to test a team of vehicles in a highly active refueling scenario. It was reduced to a 2 hour payload after experiments confirmed that any less of a payload would cause too many critical refuels. The next two reduce battery capacity simulations show how the refuel algorithm in conjunction with the token passing scheme work. The last scenario tests connectivity as well as refueling behavior between vehicles. The simulated refueling results in a vehicle leaving its specified survey zone to travel to the refuel point. Upon reaching the refuel point, the battery model instantly increases its charge to 100 percent. The vehicle's modeled typically take approximately 4 hours to recharge. With this constraint, simulating a vehicle being replaced by another vehicle was appropriate. The semantics of dynamic vehicle swapping falls out of the scope of this work and thus an "instant recharge" was used in its place. The formation of the team depends on the number of vehicles that leave to refuel. The experimentation above allows only 2 vehicles at a time (at most) to leave to refuel. The formations possible are a 6-vehicle formation, a 5-vehicle formation, which causes the center zone of tier 3 to be removed and the two outer zones to split the remaining space, making the respective vehicles shift closer to one another, and a 4-vehicle formation, which is the same as a 5-vehicle formation, but with only one vehicle in tier 2, (moved into the center of the tier). Each vehicle surveys a zone using a bow-tie pattern. This bow-tie pattern is 200m2. The three formations and bow-tie pattern can be seen in Fig. 9.

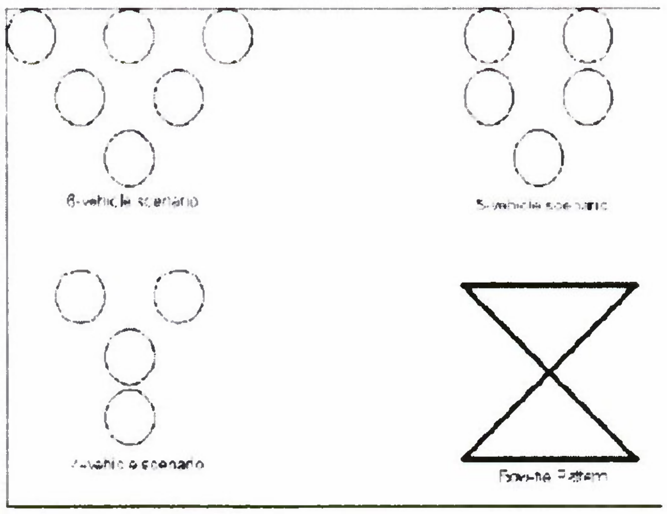

Fig. 9. Vehicle formations and their bow-tie pattern.

B. Extensive Testing of the Refuel Behavior

Extensive testing was run on the refuel behavior utilizing 0, 1, 2, and 3-token scenarios. Three tokens allows the group to reconfigure to a 3-vehicle formation, which removes the above tier. See 10 for the 3-vehicle scenario. Testing done with no tokens allows every vehicle to leave the survey area. The purpose of this scenario was to determine if utilizing the refueling behavior without the aid of token passing is a viable solution. The experiments explained in the previous section were preliminary results, showing that a token passing scheme is possible in the underwater domain. The experimentation explained below is to extensively stress test the refuel and token passing algorithms to determine their worth in the underwater domain. After some initial testing, it was shown that a vehicle with too high of an RoC would estimate negative percentage values for its energy, thus it was determined that the 2hr. battery capacity utilized in the preliminary testing was too low. To determine the appropriate battery capacity, each vehicle was placed at the waypoint in their respective bowties furthest from the refuel point. The time was computed for each vehicle to leave their waypoint, reach the refuel point, and return to their waypoint. These times were added and the total time as a result became the battery capacity. This total time was approximately 5 hours, based on the speed of the vehicles and their distances from the refuel point. The experiments are summarized below:

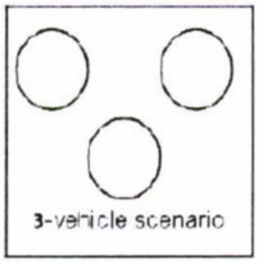

Fig. 10. 3-vehicle formation.

TABLE III
EXTENSIVE TEST SCENARIOS.

Scenario	Energy	# of Tokens
1	100%	0
1	81%	0
1	86%	0
2	Random	0
2	Random	0
3	100%	1
3	93%	1
3	86%	1
4	Random	1
4	Random	1
4	Random	1
5	100%	2
5	75%	2
5	90	2
6	Random	2
6	Random	2
6	Random	2
7	100%	3
7	85	3
7	79	3
8	Random	3
8	Random	3
8	Random	3

VI. RESULTS, CONCLUSIONS, AND FUTURE WORK

A. Results

1) Connectivity and The Basic Refueling Behavior: The first test scenario was to determine the connectivity between vehicles. Refueling was disabled, and the token was passed from vehicle to vehicle to determine if a loss would occur. The experiment showed that a token can be passed from vehicle to vehicle in a ring-like manner when the vehicles are not oriented in an actual ring. The relay vehicle successfully passed the token from the last vehicle to the first on multiple experiments. During testing, reconfiguration occurred to maintain coverage within the search area. Fig. 11 illustrates this behavior. As two vehicles, (B and E) leave to refuel, Vehicle's F and

D move in to maintain the search area left open by E's departure. Similarly, C shifts to the center of its tier to maintain the search area left open by B's departure.

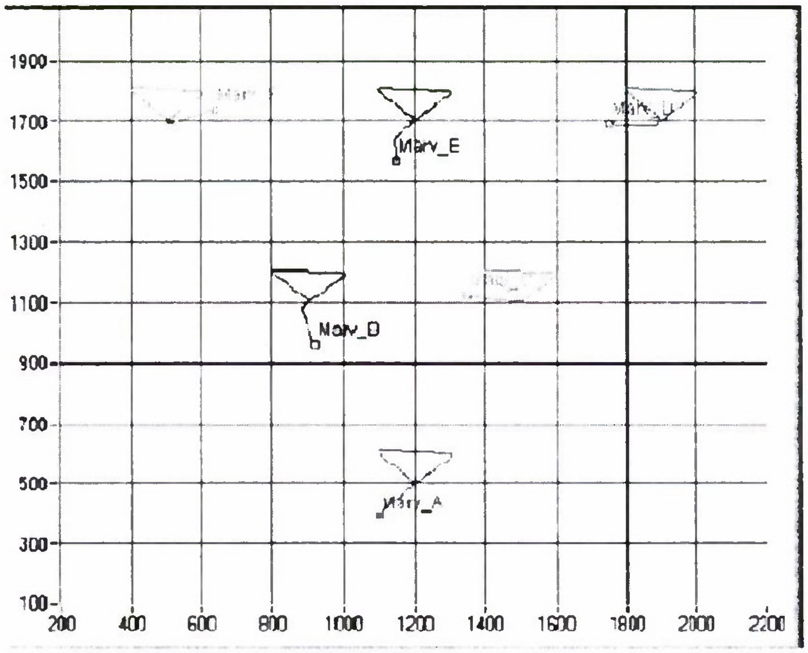

Fig. 11. Reconfiguration with 2 tokens.

An experiment to determine the appropriate reduced battery capacity was done with a payload as small as 30 minutes to one hour (to drain the energy level to critical levels). Scenario 2 was the first experiment to do this. One token was used and the two vehicles furthest from the refuel point (1200, 0, 0 in these scenarios) reached critical energy levels. To determine if the critical refuel was a result of too few tokens and not an increased rate of consumption, an additional token was added. The same issue occurred, where the two vehicles farthest from the refuel point hit critical levels. Fig. 12 illustrates the issue with slow-moving vehicles using extremely small energy payloads.

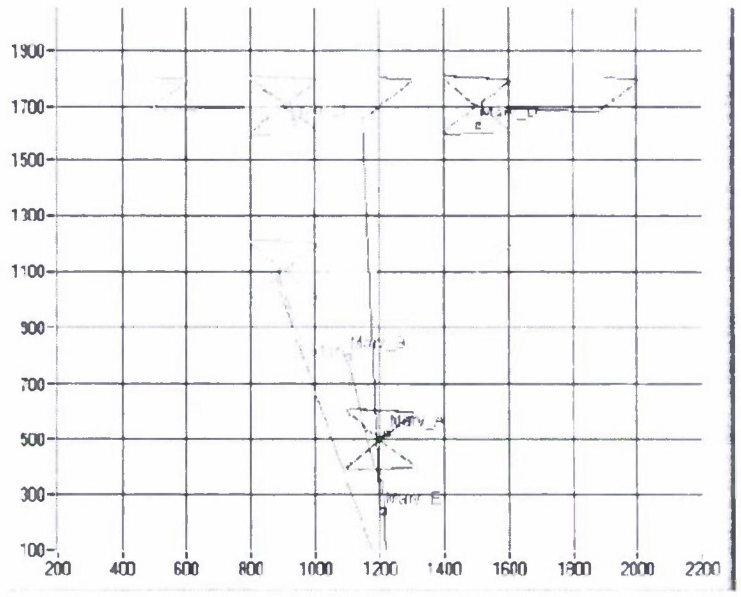

Fig. 12. Double Critical Refuel.

Fig. 12 shows two vehicles returning from refuel, (B and E), one vehicle who received the token (C) going to refuel, and two vehicles, (D and F) hitting the critical refuel threshold and leaving to refuel. This figure is from scenario 3. The grid lines are 200m apart from one another. The acoustic range is set to 750m, confirming that, out of all of the vehicles leaving their survey pattern, only C could have received the token. The speed of all vehicles was 1.25 m/s; this causes the lengths of F and Ds trails coming out from their patterns to signify that they reached a critical refuel several minutes after C had already left its survey pattern. Cs critThresh is lower as it is closer to the refuel point than D and F, further confirming that C did not enter a critical refuel. After running this test, the payload was set to 2 hours for the next two experiments. Fig. 13 illustrates scenario 5. One of the returning vehicles (B) passes the token (to C) while moving to its position. The direction B is heading indicated that as it was traveling to the consensus point, (center of the survey area), it gained consensus, and immediately began moving to its new position, passing the token to the next vehicle.

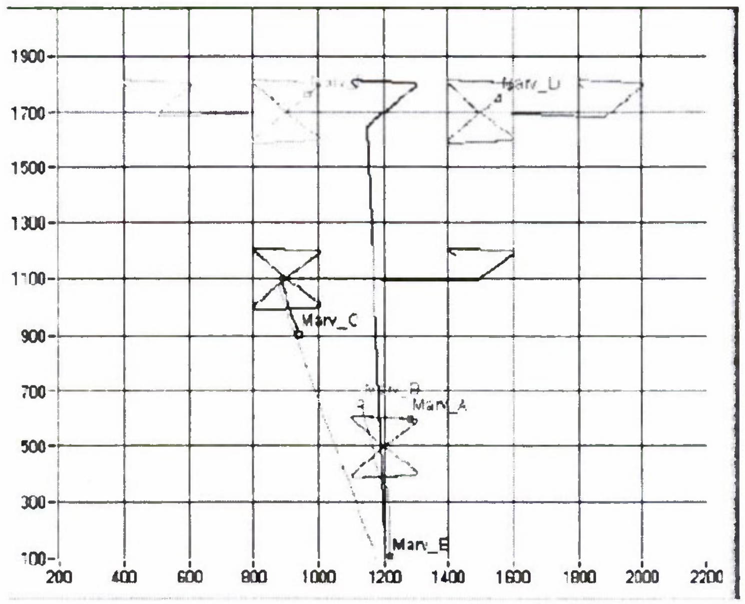

Fig. 13. Vehicle passing the token while moving into position.

In a test run in scenario 3 (not shown), the vehicle, (C in this case), would pass the token before gaining consensus, (this behavior is indicated by the vehicle which received the token leaving its survey pattern while the returning vehicle was still traveling to the consensus point). This can only occur if the vehicle passes the token before gaining consensus. The only way a vehicle can pass the token is if all flags signifying either a refuel or a rejoin are set to false. The false-flags signal that the vehicle has consensus. Timing regarding the setting of the refuel flag to false and the rejoin flag to true caused the vehicle to pass the token before it had gained consensus. This was fixed for later experimentation but it should be noted that the 6-vehicle, 2-token scenario represented in Fig. 13 did not have this fix, confirming the issue was indeed a timing problem. After the reduced battery capacity testing was completed, one final experiment in Scenario 6 (not shown) was run to test the connectivity of the vehicles both while the token is being passed and while a vehicle is out to refuel. Critical refueling was not an issue in this experiment due to the large battery capacity of all of the vehicles. The experiment successfully showed a token making a full circle between vehicles multiple times as well as allowing a vehicle to refuel and the subsequent team reconfiguration.

2) Extensive Testing of the Refuel Behavior: While the preliminary testing has shown that a token passing scheme can be utilized by a behavior in the underwater domain, extensive testing was needed to test the viability of such an approach and what measures were needed to ensure effective use.
A set of preliminary tests were run for Scenario 1 using the refueling algorithm with prediction activated. These tests confirm that when allowing a vehicle to leave of its own accord with no token blocking mechanism while having the same energy level as its teammates, all vehicles will leave. The only deterrent for a vehicle leaving when there are no tokens available is the confirmation system. A vehicle must gain acknowledgement from another vehicle when it is leaving to refuel. The testing has shown that 5 vehicles requested refuel within 5 minutes of one another to refuel. 3 of these vehicles requested refuel within 3 seconds of one another. The

quorum for this work is 3. refueling vehicles Any number of vehicles above 3 is considered a catastrophic failure. Fig. 14 and Fig. 15 show this.

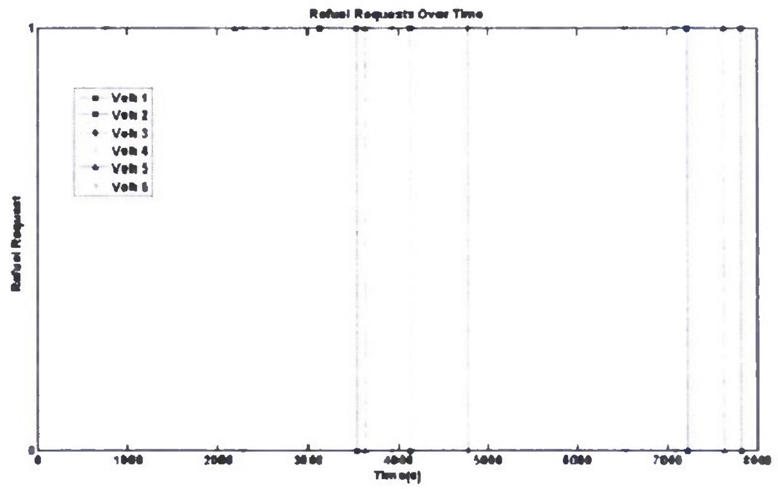

Fig. 14. Representation of each vehicle's refuel request in Scenario 1. 1indicates a request, 0 indicates a return.

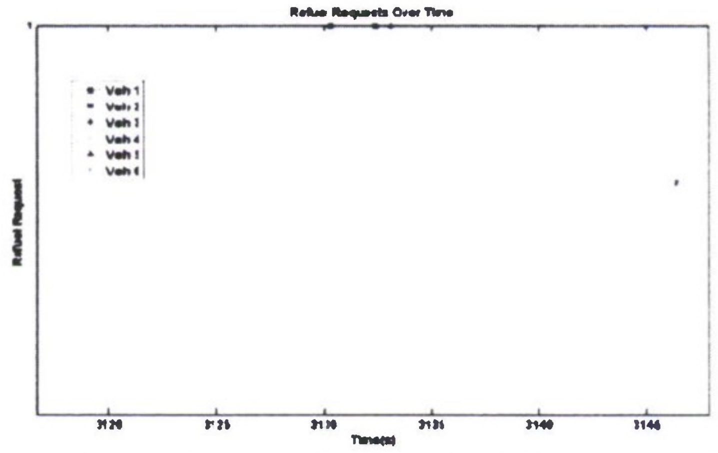

Fig. 15. Zoomed in view of the refuel request in Scenario 1. 3 vehicles request to refuel within 5 seconds of one another.

Testing in Scenario 2 shut off the prediction of the vehicle but left on the dynamic refuel threshold. This allows each vehicle to take into account other vehicles which it thinks will either leave or have left to refuel. This shows that by randomizing each vehicles energy, a natural staggering effect occurs. Testing did not show more than 3 vehicles leaving to refuel. This data shows the possibility of a vehicle conforming to the quorum imposed by this work, but as there is no actual way to impost the quorum, it is not guaranteed that the quorum will always be maintained. Regardless of this fact, Fig. 16 and 17show the possibility of refueling to occur within acceptable boundaries without the need of a token.

Fig. 16 shows the various states of the vehicles, between being in transit, to refueling to patrolling. The importance of this figure is the presence, (or lack) of a line going across the 1 position. This line represents vehicle coverage. No line indicates that no vehicle is in their patrol position. This implies that no vehicle is currently explicitly patrolling their area, rather, they are transiting to either the refuel point or another area. A transiting vehicle is a reduction in coverage not a lack, but by using this as a baseline, measurements in the length of these gaps represent improvements of the algorithm when the token is present. Fig. 17 shows each vehicle's energy level, refuel threshold, and critical threshold. The critical threshold played no part in the decision of the vehicle (rather than acting as a bound for the refuel threshold), as the vehicle's energy never got low enough for it to be relevant. Scenarios 3 and 4 have 1 token, indicating that all 6 vehicles are part of the same sub-team. The data in Fig. 18 shows a drop in the "no coverage time" of the group. The data also shows a trend that will occur throughout all the testing. As the number of token increases. blocks of reconfiguration begin to form rather than the configuration being scattered about throughout the test. Fig. 18 shows the reconfiguration to be spread throughout the test, indicating that overall coverage is slightly lower throughout the entire test, but relatively uniform. Fig. 19 shows the order at which each vehicle left to refuel. The order being d, e, f, a, b, c.

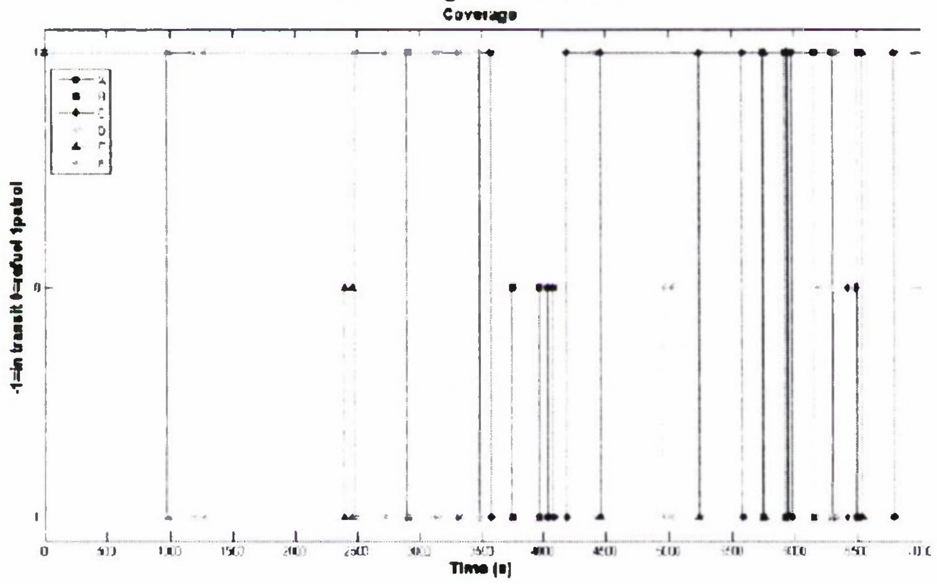

Fig. 16. The life of the vehicles during the mission. -1 represents a vehicle in transit, 0 represents a vehicle returning from refuel, and 1 indicates a vehicle patrolling its designated area.

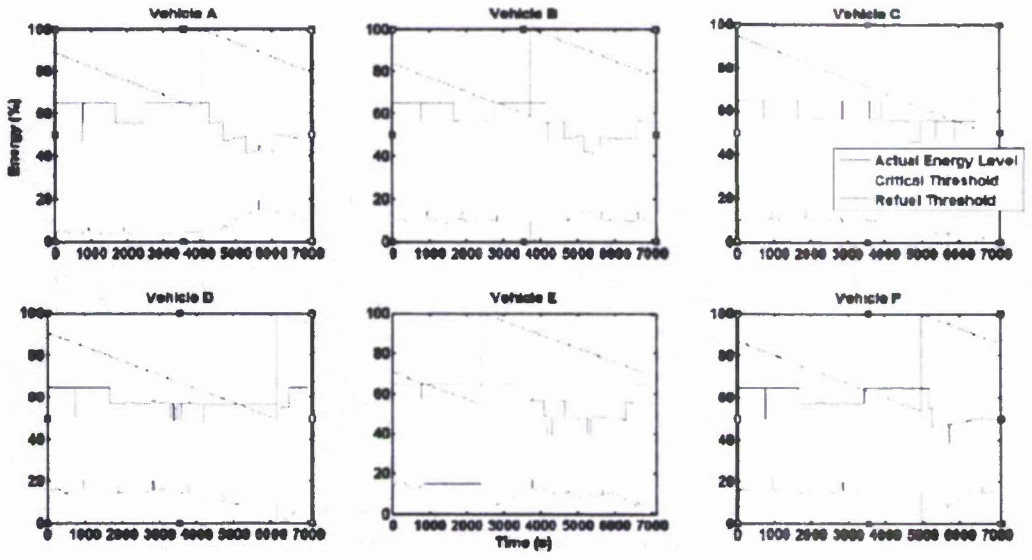

Fig. 17. energy data on each vehicle in Scenario 2. These graphs show the vehicle's energy level, refuel threshold, and critical threshold.

The spike in the vehicle's actual energy level indicates a refuel event. This figure also shows the vehicles refuel threshold and critical threshold, but also shows the vehicles predicted energy level (the value computed based on the number of vehicles in its computation and where they are perceived to be located), as well as when the vehicle received the token (circle), and when the vehicle through it would receive the token next (square). Testing has put the prediction in energy level to be at +/- 1% from the actual and the vehicles prediction of next token reception to be within 5 minutes of the actual reception when no vehicles are present in the computing vehicle's algorithm. The prediction, by design, becomes conservative when vehicles are added and can drop drastically. This is used to allow the current vehicle with the token the opportunity to leave early and avoid a critical refuel.

Scenarios 5 and 6 each contain 2 tokens, splitting the 6 vehicle group into 2 sub teams, a, b, and c and d, e, and f. This testing had extensive loss of tokens (5 of 6 tests run had at least one lost token), indicating that the relay vehicles could not pass the token to the recipient. This is thought to be due to the way in which the vehicles reconfigure during the tests, though it could not be confirmed. The data in Fig. 20 shows the transition between spread reconfiguration and grouped reconfiguration.

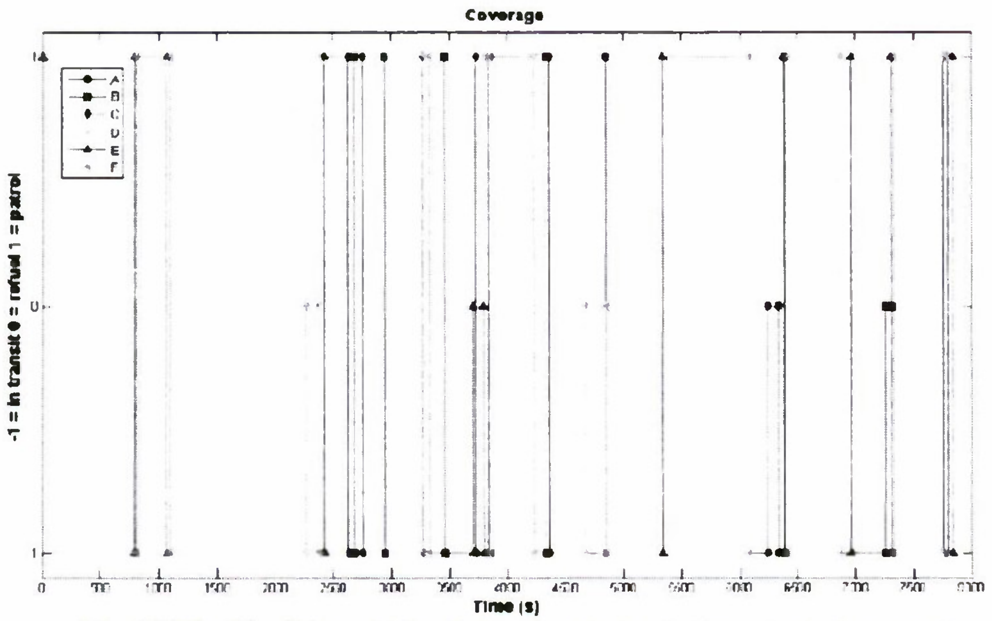

Fig. 18. The life of the vehicles during the mission in Scenario 4. -1 represents a vehicle in transit, 0 represents a vehicle returning from refuel, and 1 indicates a vehicle patrolling its designated area.

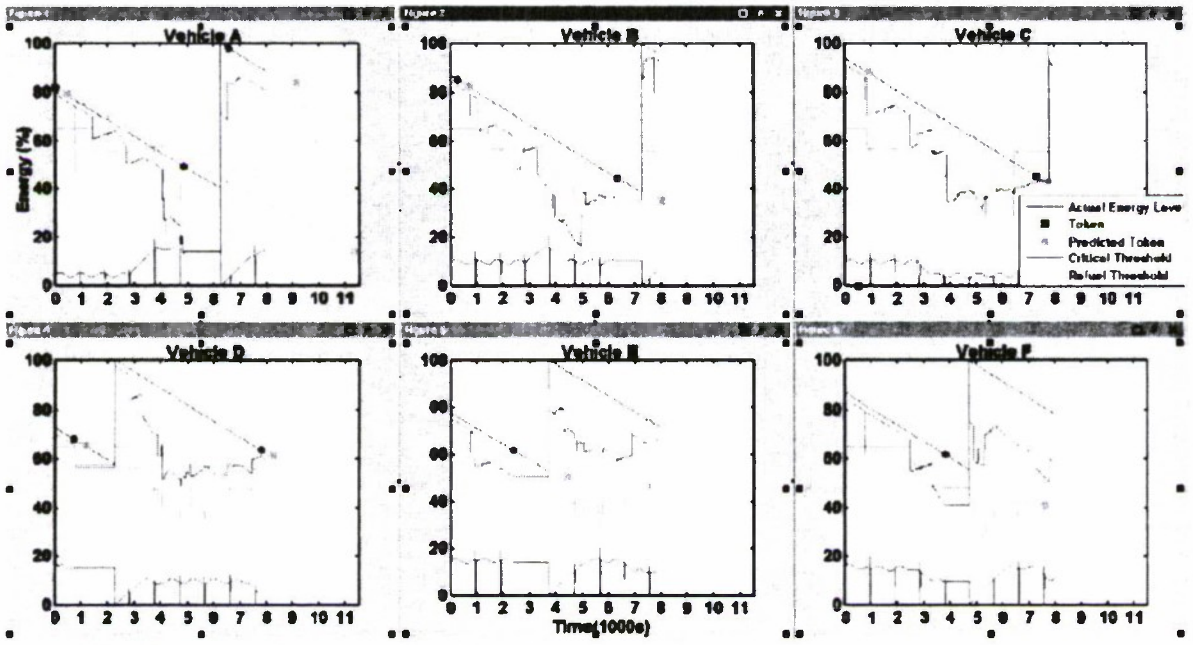

Fig. 19. energy data on each vehicle. These graphs show the vehicle's energy level, refuel threshold, and critical threshold. This data was from Scenario 4; each vehicle's energy level is a random number between 70 and 100%.

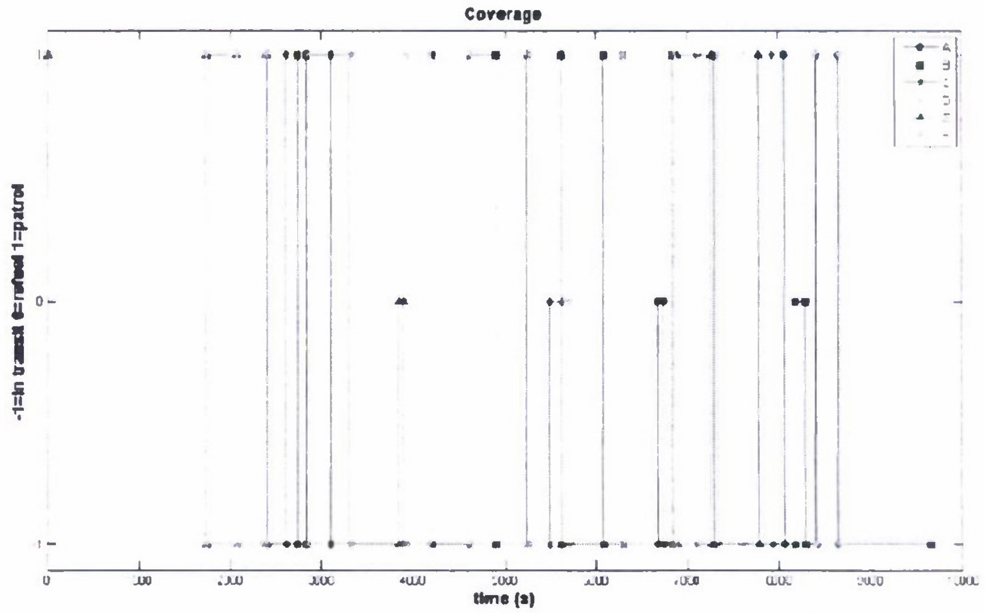

Fig. 20. The life of the vehicles during the mission in Scenario 6. -1 represents a vehicle in transit, 0 represents a vehicle returning from refuel, and 1 indicates a vehicle patrolling its designated area.

This data also shows a slight decrease in the "no coverage time" of the group, approximately 600s (data after 8300s is discarded due to lost token. The lost tokens occurred at approximately 5500s and 8300s respectively) as opposed to the Scenario 2's 800s, though there is an increase in the gap from Scenario 4's 400s. Fig. 21 shows each vehicle's data. The order of initial refuel was b, e, c and d, and lastly a. Vehicle F never got a chance to refuel due to a lost token. There is an addition to the data represented by a thick line. This line represents the number of vehicles represented in the prediction and refuel threshold algorithms. This line was added to correlate the movement of the refuel threshold with the number of vehicles perceived to refuel. The maximum number of vehicles that can be added to each vehicle's computation is dependant on their sub-team, not the group as a whole. The line shows how adding a vehicle can cause drastic drops in the refuel threshold as well as the predicted energy level. The fluctuations shown in the predicted energy level while the number of vehicles present in the algorithm was stable was a result of movements of all vehicles within the operations area. As distance changes, the estimate of the predicted energy level will change as well. Fig. 21 also shows an increase in number of times a token was received due to the availability of the token. Scenarios 7 and 8 each contain 3 tokens, splitting the vehicle group into 3 sub-teams, a and b, c and d, and e and f. Testing with 3 tokens had no lost tokens in any of the 6 tests run. Fig. 22 shows a further grouping of the vehicle's reconfigurations. There is a noticeable difference between the reconfiguration spread in this figure, and the spread in Fig. 18. The "no coverage time" is also reduced from that in Fig. 20 by approximately 100s, indicating that the grouping effect helps the overall connectivity of the group. Fig. 23 shows a reduction in refueling among vehicles as opposed to Fig. 21. This reduction is due to the availability of the token, but unlike the previous two scenarios, as a result of the token being made available more often, vehicles had less of a need to use it as their predictions on future energy were more stable (shown by the long tracks of straight-line predicted energy graphs for each vehicle).

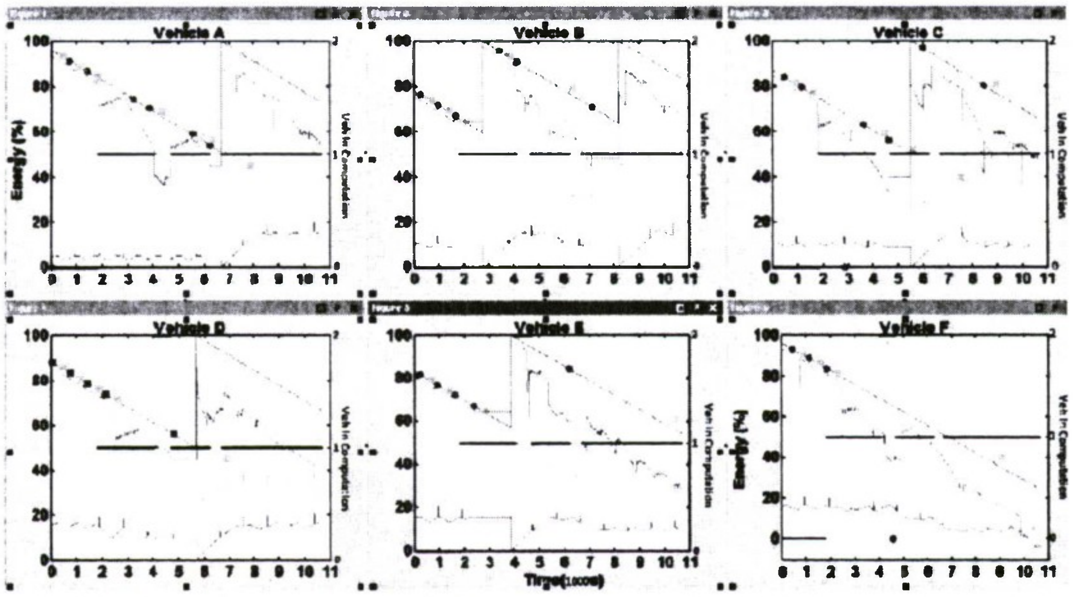

Fig. 21. Energy data on each vehicle. These graphs show the vehicle's energy level, refuel threshold, and critical threshold. This data was from Scenario 6; each vehicle's energy level is a random number between 70 and 100%.

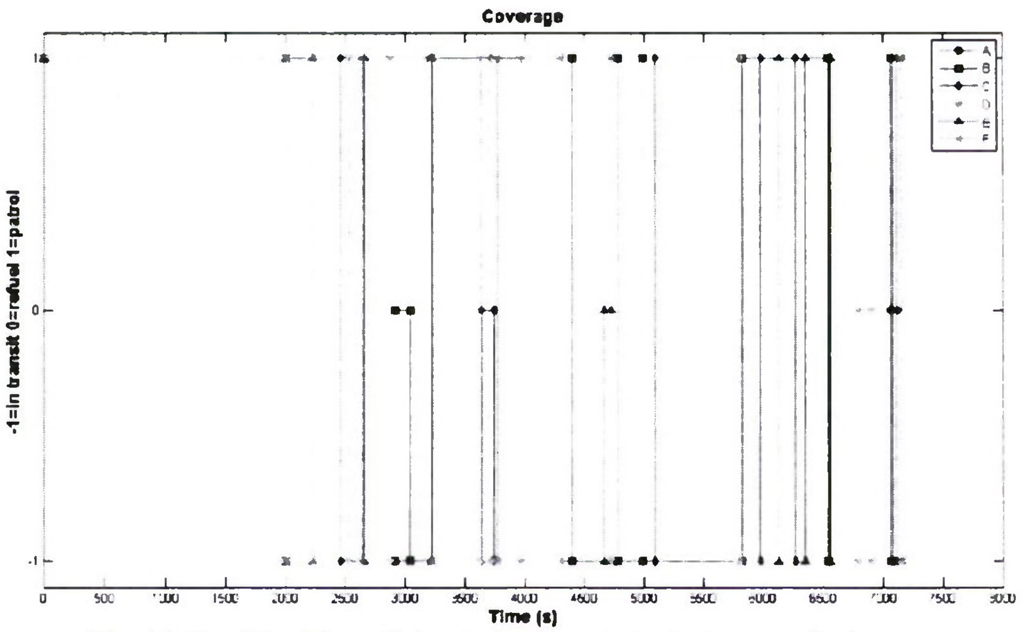

Fig. 22. The life of the vehicles during the mission in Scenario 8. -1 represents a vehicle in transit, 0 represents a vehicle returning from refuel, and 1 indicates a vehicle patrolling its designated area.

This figure also shows the major increase in the amount of occurrences when each vehicle received the token. This is to be expected as each token was only shared between two vehicles.

B. Conclusions

The testing confirmed the effectiveness of using a token-passing refueling scheme in 24-hour mission scenarios involving a team of Autonomous Undersea Vehicles (AUVs). Multiple experiments confirmed the connectivity of the vehicles by showing a token being passed throughout the team, from first to last and back to first, as well as the ability of the team to comprehend the need of a vehicle to refuel and the subsequent change of values (from token ownership values to refueling values) and act accordingly. Extensive testing has shown that the number of tokens, starting energy levels, as well as the vehicle reconfiguration play a major part in how well the group can maintain coverage in the survey space.

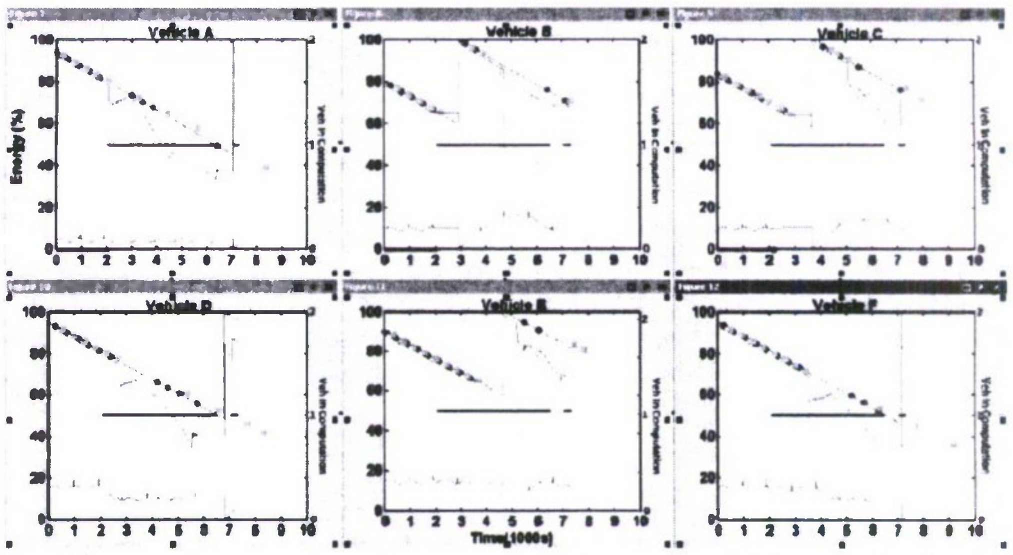

Fig. 23. energy data on each vehicle. These graphs show the vehicle's energy level, refuel threshold, and critical threshold. This data was from Scenario 8; each vehicle's energy level is a random number between 70 and 100%.

Testing has shown that, with the group size used, 1 token will allow for general stability, i.e. having the majority of vehicles maintaining a larger coverage area throughout testing, though reconfiguring occurs regularly, while 3 tokens allows for periodic stability, i.e. reconfiguration will occur frequently in large packets, though there will be periods of no reconfiguration as a result of multiple vehicles having refueled simultaneously. Testing has also shown that refueling can decrease to levels equivalent to 1 token when multiple tokens are used. This is a result of the availability of the token as well as the number of vehicles the token needs to be shares with. These factors cause the predicted energy levels to be more stable and stay above the refuel threshold, indicating less of a need to use the token. The token passing scheme used a token ring. This scheme forces the token to be passed sequentially from the vehicle with the lowest ID to the vehicle with the highest ID, repeating this pattern. When the vehicles are not in order or, or at least able to directly contact the recipient vehicle when sending a token, a relay vehicle is necessary to pass along the token to its intended recipient, or at least pass the token closer to its intended recipient by relaying the token to another relay vehicle. This relay vehicle can become a single point of failure, causing the token to be lost. The majority of the testing had little to no lost tokens, but the 2 token scenarios had lost almost all of the tokens present over all tests. Some tests had one lost token, some had both. It is not clear exactly why the 2 token tests had more lost tokens that any other test, though it is hypothesized that, as a result of the team having the ability to be spread out over multiple tiers, many relay vehicles were used, causing the token to be lost. Not insuring complete connectivity between sender and receiver can have a negative impact. When using a token, particular care needs to be taken to ensure that the recipient can actually receive the token. While the token-passing scheme has great potential, unless the vehicle formation is an actual ring, utilizing a ring configuration for message passing is inappropriate due to the potential for loss and confusion among teammates, regardless of the presence of relay vehicles. A token-passing configuration that can ensure connectivity between sender and receiver is essential to the effective use of token-passing. This connectivity allows for direct acknowledgements from the receiver, removing the need for relay vehicles, as the only vehicles necessary to know the current state of the token are the sender and receiver of the token. Passing a critical piece of information using a medium prone to noise and loss involves many fail-safes to ensure effective use. Situations involving loss of the token can occur even in the most cautious of scenarios and operator interaction may still be required. The point of this research is not to show a superiority of this particular scheme, but to solve a very common problem using few resources (message space and memory on-board the vehicle, in this case), in a simplistic manner and among a low-bandwidth, noisy communications network. All of the experimentation was done using an acoustic modem simulator that simulated whether or not a vehicle could hear another vehicle using spatial distance, (in these experiments, 750m was the maximum distance two vehicles could be from one another to still receive information). Further testing can be done using a more accurate modeling of the acoustics.

C. Future Work

The utilization of an energy management system such as this opens up many opportunities to tie in navigational behaviors which take advantage of the available sensory data to determine a best path scenario for reduction of energy consumption, or to decrease the time to a waypoint. By creating a module that utilizes a modified shortest path algorithm which uses energy as a bound, ocean current information read from sensors onboard can be combined with the energy level of the vehicle to create an optimized path of travel. [31] and [32] have studied the addition of oceanic current information in navigation of a UUV. [31] focuses on the appropriate modeling of an ocean current sensor while [32] implements a horizontal Acoustic Doppler Current Profiler (H-ADCP) in the navigation of an AUV to reduce travel time to a waypoint. Work is currently being proposed to expand the token passing scheme into a general distributed task management scheme. The idea is to utilize the token passing scheme for any task that cannot be done either in parallel with other tasks or can utilize command fusion, (such as obstacle avoidance and waypoint following). The token passing scheme would need to be modular and implement a simple interface to different tasks. This simple interface would allow the token passing scheme to work on potentially any system with any task. In order to make the scheme robust, communications needs to be maintained. By maintaining a level of connectivity between neighboring zones and forcing acknowledgements, the token passing using a relay vehicle as mentioned in section 3 can be removed entirely. Instead of using a token ring, a different token-passing configuration, like a token ladder is used. A token ladder is a token passing scheme where instead of the last node sending the token to the first, the last node sends the token to the node just below it. The token acts as if on a ladder, (up and down from first to last and back again). By passing the token up and down in a ladder movement, the nodes can be independent of the token. The token would be passed from zone to zone rather than from vehicle to vehicle. Because zones are connected to their neighbors, a relay is no longer necessary. The implementation of this scheme would need to take into consideration time delay when the token is being sent above or below the node. Testing for this work was done using randomized energy levels. In a real-world scenario it is unlikely that one would place a vehicle with less than full charge out to perform a mission. To gain the benefits of the staggering effect that randomized energy levels produces, each vehicle could further modify their refuel thresholds based on position. This position modification would cause vehicles that would once wait, to refuel based on their distance away from the refuel point. This effect would also cause the minimum refuel threshold (the threshold that occurs when all teammates are present in the algorithm) to be different on each vehicle, allowing the staggering effect to be present at all times.

REFERENCES

[1] P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe, "Token approach for role allocation in extreme teams: analysis and experimental evaluation," 2004.

[2] M. Benjamin, P. Newman, H. Schmidt, and J. Leonard, An overview of MOOS-IvP and a Brief Users Guide to the IvP Helm Autonomy Software. Massachusetts Institute of Technology, Cambridge, MA, USA, June 2009.

[3] D. of The Navy, The Navy Unmanned Undersea Vehicle (UUV) Master Plan. Department of The Navy, USA, Nov, 2004.

[4] M. Stojanovic, "Underwater acoustic communication," Wiley Encyclopedia of Electrical and Electronics Engineering, 1999.

[5] H. Nguyen, S. Giordanot, and A. Puiatti, "Probabilistic routing protocol for intermittently connected mobile ad hoc network* (propicman)," in WoWMoM, IEEE International Symposium, (Switzerland), ISIN-DTI, SUPSI, 2007.

[6] Z. Li and H. Shen, "Utility based distributed routing in intermittently connected networks, (udm)," in ICPP, 37th International Conference on Parallel Processing, (Fayetteville, AR 72701Fayetteville, AR 72701), University of Arkansas Department of Computer Science and Computer Engineering, 2008.

[7] P. Xie, J. Cui, and L. Lao, "Vbf: Vector-based forwarding protocol for underwater sensor networks," University of Connecticut, University of California at Los Angeles, 2006.

[8] Z. Guo, G. Colombit, B. Wang, J. Cui, D. Maggiorinit, and G. Rossit, "Adaptive routing in underwater delay/disruption tolerant sensor networks," 2008.

[9] N. Nicolaou, A. Seet, P. Xie, J. Cui, and D. Maggiorini, "Improving the robustness of location-based routing for underwvater sensor networks," in MTS/IEEE OCEANS Conference, (Storrs, CT 06029, USA and via Comelico 39, 20135 Milano, Italy), Computer Science and Engineering Department, University of Connecticut and Computer Science Department, University of Milano, June 2007.

[10] G. Xie, J. Gibson, and L. Diaz-Gonzalez, "Incorporating realisticacoustic propagation models in simulation of underwater acoustic networks: A statistical approach," in OCEANS 2006, (Monterey, CA 93943, USA), Naval Postgraduate School, 2006.

[11] M. Ayaz and A. Abduallah, "Underwater wireless sensor networks: Routing issues and future challanges," in Conference on Advances in Mobile Computing and Multimedia, (Kuala Lampur, Malaysia), CIS Department, Universiti Teknologi PETRONAS, 2009.

[12] D. Eickstedt and M. Benjamin, "Cooperative target tracking in a distributed autonomous sensor network," 2006.

[13] D. Eickstedt, M. Benjamin, and H. Schmidt, "Adaptive control of heterogeneous marine sensor platforms in an autonomous sensor network,"

[14] I. Akyilidiz, D. Pompili, and T. Melodia, "State-of-the-art in protocol research for underwater acoustic sensor networks," in WUWNet, (Atlanta, GA, USA), Georgia Institute of Technology, September 2006.

[15] D. Pompili and I. Akyildiz, "Overview of networking protocols for underwater wireless communications," IEEE Communications Magazine, January 2009.

[16] M. Alighanbari and J. How, "Robust decentralized task assignment for cooperative uavs," in AIAA Guidance, Navigation, and Control Conference and Exhibit, (Cambridge, MA, USA), Aerospace Controls Laboratory Massachusetts Institute of Technology, August 2006.

[17] E. King, Y. Kuwata, M. Alighanbari, L. Bertuccelli, and J. How, "Coordination and control experiments on a multi-vehicle test bed," (Cambridge, MA, USA), Aerospace Controls Laboratory, Massachusetts Institute of Technology, July 2004.

[18] L. Parker and F. Tang, "Building multirobot coalitions through automated task solution synthesis," 2006.

[19] B. Gerkey and M. Mataric, "Pusher-watcher: An approach to fault-tolerant tightly-coupled robot coordination," (Los Angeles, CA, USA), Computer Science Department, University of Southern California, May 2002.

[20] M. Dias and A. Stenz, "Traderbots: A market-based approach for resource, role, and task allocation in multirobot coordination," August 2003.

[21] P. Ulam, Y. Endo, A. Wagner, and R. Arkin, "Integrated mission specification and task allocation for robot teams - part 1: Design and implementation," 2006.

[22] P. Ulam, Y. Endo, A. Wagner, and R. Arkin, "Integrated mission specification and task allocation for robot teams - part 2: Testing and evaluation," 2006.

[23] A. Farinelli, Distributed Task Assignment for Real World Environments. PhD thesis, Rome, Italy, 2005.

[24] G. Martel, "Distributed control environment (dice) users guide.".

[25] SCIS, IEEE Standard for Distributed Interactive Simulation-Application Protocols. The Institute of Electrical and Electronics Engineers, Inc., New York, NY, USA, 1995.

[26] P. Newman, MOOS - Mission Oriented Operating Suite. Massachusetts Institute of Technology, Cambridge, MA, USA, 2001.

[27] M. Benjamin and H. Schmidt, A Guide to the IvP Helm for Autonomous Marine Vehicle Control. NAVSEA Division Newport, Dept. of Mechanical engineering, Massachusetts Institute of Technology, Newport, RI, USA, Cambridge, MA, USA, November 2007.

[28] T. Schneider and H. Schmidt, MOOS-IvP Communication SoftwareStack Users Guide. Laboratory for Autonomous Marine Sensing, Department Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA, USA, Jan 2010.

[29] I. Suzuki and T. Kasami, "A distributed mutual exclusion algorithm," ACM Transactions on Computer Systems (TOCS), v.3 , Issue 4, November 1985.

[30] K. Raymond, "A tree-based algorithm for distributed mutual exclusion," ACM Transactions on Computer Systems (TOCS)v.7 , Issue 1, Febuary 1989.

[31] N. Vasquez and M. J. Rendas, "Using a priori current knowledge on auv navigation,"

[32] B. Garau, A. Alverez, and G. Oliver, "Auv navigation through turbulent ocean environments supported by onboard h-adcp," (Balearic Islands, Spain), IMEDEA Research Center, University of the Balearic Islands, May 2006.