Distributed Adaptive Sampling using Autonomous Surface Vehicles (ASVs)

Anthony Matos, Ryan Mcguire, Akshat Shah
Supervisor: Mehdi Rahmati, Professor Dario Pompili

ABSTRACT

There is a need for an automated system of robots that investigate the aquatic environment in the most dynamic and seamless fashion possible - adaptive sampling is the answer. Adaptive sampling focuses on the implementation in which the machines performing the practice investigate a small area of interest and makes decisions on its own to proceed to the next area of interest until it finds a kind of data it is targeting. Ideally, in an aquatic setting, a team of multiple robots are supposed to get together and communicate the data they read and decide on the area of interest without any human input. Included in this project are procedures of adding multiple machines under the same network with an efficient communication system and feeding the machines with behaviors of investigating aquatic data with attached sensors and react to those data to display adaptive sampling behaviors.

Terms: Nodes - machines to add under network, BlueRov2/Pixhawk - machine of interest for the experiment, BATMAN-ADV -Better Approach To Mobile Adhoc Networking - Advanced

I. Introduction

Adaptive sampling is a new improved solution focusing on the real-time acquisition of aquatic data. It outshines obsolete and inefficient methods for underwater reconnaissance. These past methods incorporate a static sensor network in which the underwater vehicles are configured to follow a fixed/random trajectory with a less clear objective. Blind reconnaissance of aquatic abnormalities or variables is very ineffective in that areas of scientific interest that are subject to change at any time by occurring abruptly or changing without notice. A prerequisite skill essential to their functionality is to be able to share the different tasks at hand in real-time to other autonomous vehicles to ensure that the collaborative work is efficient and that energy consumption is optimal. For example, it is ideal to have some vehicles relay to others a specific task such as the measure of pH concentration such that the other vehicles can prioritize unvisited tasks and not waste time and energy.

Ocean weather forecasting is one application that would benefit from the existence of autonomous underwater vehicles in that it deals with searching vast underwater bodies and is relevant in learning about the environmental changes caused by any abnormalities that are undetectable by humans alone. The use of autonomous underwater vehicles will help provide more insight into the underwater unknowns and will also become a major asset to biologists and oceanographers. This collective union of multi-tasking robots can make up for what these aquatic scientists lack. Great potential lies within the communication aspect of said vehicles if only there was a leader-less topology of network communication among them on the surface. Specifically, for any number of autonomous vehicles, any subset of them should be able to freely and directly communicate with other vehicles without needing a middleman in a sense to pass on data relating to temperature, water pollution, dissolved oxygen, pH, turbidity, and algae concentration. It must be able to detect harmful changes and their effects on our ecosystem.

The under observed aquatic environment on earth needs a solution to be able to pinpoint problem areas in real-time and be able to draw connections. This can be done with multiple autonomous vehicles, given areas of interest and wireless communication between each other will enable the aquatic environment to be sampled with cost and time in mind. This team of vehicles enabled with distributed adaptive sampling will save time and money. This is done in two phases, Phase 1 and Phase 2. These are in a way to prevent losing valuable data by solely basing it off of past data on important areas. Phase 1 is basically for the team of robots to navigate the area and get an idea of the environment. This is done every time for a certain area that is to be observed like a river or pond, or as needed as the oceanic environment is ever-changing. Then Phase 2 goes underway, which is where adaptive sampling comes into place where there are more samples taken in areas that show more change in levels that are being sampled, such as pH, temperature, or salinity. Results show that this way of sampling is faster and more efficient than the current ways of sampling.

The whole premise of adaptive sampling is to construct a formation that is resilient to geometric constraints after the first sweep of underwater qualities. After one sweep, the robots are to determine from the sampling data the best trajectories to pursue that would lead to faster acquisition of data. Once the AUVs have a better understanding of where the data is, there is no need to pass through the same areas again. Adaptive sampling is used to learn about one or more characteristics of the population of interest by investigating a subset or sample of the desired population [10]. Once the sample is investigated, the population quantities of interest are obtained. Used for statistical experiments, adaptive sampling obtains data by adjusting the experiment as needed. In a non-adaptive environment, decisions and attributes are kept static from

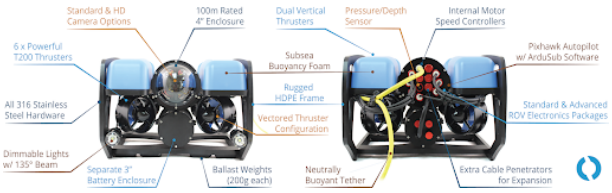

Fig. 1: Visual image of the vehicle that will be used for adaptive sampling

the beginning. However, in adaptive sampling, there is more of a dynamic approach as the decisions and attributes keep adjusting themselves as the experiment undergoes continuous interactions.

The underwater vehicle used is the BlueRov2 which is fitted with raspberry Pi to fit an autonomous functionality that would enable our adaptive sampling to work. The Pixhawk 4 controller is paired with the BlueRov2 such that it can send signals to the electronic speed controllers to send power to the various motors. This is done via pulse width modulation, or PWM, which is a very common method of power distribution in robotic vehicles. The many propellers needed to navigate the aquatic environment using several channels coming off the Pixhawk 4 controllers. The robot must activate many channels to counteract outside forces such as ocean currents. It must have many sensors to be able to sample several aspects of the ecosystem such as pH, temperature, and salinity. The onboard GPS can give the location to the server to locate the vehicle with the other ones but is only usable on the surface of the water.

Connection techniques for this robot from the factory involve a tether as shown in the diagram, but it was decided that the Pis should be connected wirelessly using an ad-hoc network. This is done via batman-adv configuring a topology such that there is no master-slave relationship that enables the free communication of any Pi to the other Pis.Once the vehicles can operate on this network topology, they should be able to communicate with one another on the tasks pertinent to adaptive sampling.

To establish adaptive sampling underwater, it is imperative to have a communication system that conducts the data in real-time. The purpose is so that the autonomous underwater vehicles (AUV) can process the data as they receive it and communicate with its team of robots to know what sample of data to focus on [13]. AUV communication is important for reprogramming and data retrieving purposes. In usual cases, the communications implemented for the AUVs have been about connecting the hardware with the serial board and retrieving that data and reprogrammed accordingly. This means that the data AUVs collect has to leave the area of analyzing data to give the current data from where they were deployed [9]. To get live data and analyze from where the AUVs are investigating the site, data must be communicated continuously to the data retrieving site and receive instructions to reprogram themselves for necessary re-investigations [7]. Radio floats and radio antennae have served the purpose of communicating the wireless connection between the AUVs, upon surfacing, and their data-centers through radio packet modem communication. Since the floats are likely to be affected by their cable limits and obstructed waters, they can not be well established [6]. Additionally, as the floats have limited projection height out of the water, they might not send the reliable quality of data or images because of lower transmission. Therefore, AUVs nowadays are generally attached to various sensors for proper location and data retrieving information in water.

II. Methodology

For robotics and computer-based experiments and projects, adaptive sampling becomes convenient by optimizing the resource allocation and parameter selection for repeated computer simulations and data communication. Since a computer can collect the data, it can study how to compile the obtained and run necessary programs for further program executions. Adaptive sampling, therefore, can have a significant implementation in machine learning(or deep learning) for a computer's active data processing capabilities.

The methodology of adaptive sampling focuses on gathering some samples and make estimations of deciding on what type of sample to gather in the next round of analyzing the data. To implement this, the experiment must be carried in a few phases. First, one should decide the objective for the type of data to find. Second, allocate samples over different areas of the experiment. Third, notice the sample that reflects the closest to our objective data. This sample now becomes our sample of interest. Moving on, more sub-allocations are made in the sample of interest and their data are studied. Continuously, the sample of interest is determined through various rounds until the data of interest is obtained. To understand it better, consider the example of finding a new streak of gold, which is our objective data. Collecting rock samples from different places and analyzing their characteristics helps to determine which place should be the sample of interest. Going to that place and studying rock samples from different areas of that place of interest for analyzing data gives a closer estimation of where the gold streak is located. Doing this process repetitively successfully results in finding the gold streak efficiently.

III. Related Works

One of the initial obstacles that must be overcome in the realm of wireless communication between numerous autonomous robots is to come up with a network topology such that any particular underwater vehicle can communicate with another directly without the need for the assistance of another. Some major common solutions to this issue are to devise some form of sampling algorithm for network analysis to customize and tailor the interaction between the AUVs to behave a certain way. The most common topologies and their associated algorithms are Ring Lattice, small world, Erdos Random, Core Periphery, scale-free, and cellular [1]. Despite their sophisticated behavioral and structural differences, their collective shortcoming is that some nodes rely on other nodes based on their relative location. There is always one intermediate node that connects two other ones from the circular ring. To their nature, they all are set up such that a directed edge

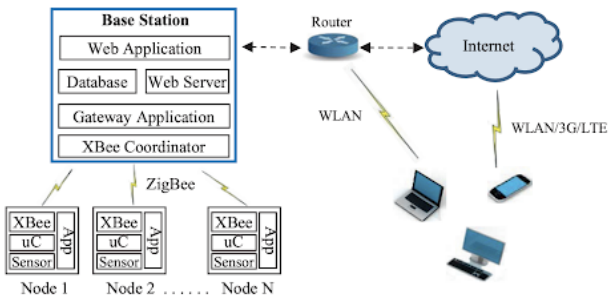

Fig. 1. Overall system architecture.

Fig. 2. (a) Raspberry Pi Model B; (b) Arduino UNO R3; (c) XBee Pro S2B; (d) RHT03 Temperature and humidity sensor.

Fig. 2: This figure represents the ad-hoc network visually to represent what will be taking place

is connected from any node to its nearest neighbor. For the applications of adaptive sampling, this is not ideal because this solution would not maximize efficiency in task allocation and energy. The more robots that have to get involved just to relay information from the sender to the receiver contributes significantly to the overhead.

One notable approach that was decided to be used in addressing this shared issue of a sought after peer-to-peer communication is the implementation of multi-hop mesh networking. However, there were various applications of its use in versatile situations. Along with ad-hoc/multi-hop configurations, one key difference in the way it was used is manifested in a specific wireless sensor network design accompanied with Arduino and raspberry Pi for environmental monitoring [5]. Such a network configuration is decentralized in that it is detached from an established infrastructure setup because it does not rely on the use of wireless access points, wired networks(Ethernet), or routers. Much of the communication has to be done in RF since wireless signals do not travel underwater. The brilliance in ad-hoc lies in its robustness and low administration costs. Peers represented by nodes can leave or join the network at any time without destroying it and hence, it highlights the fact that no group leader is playing a part in managing this connection. This robustness also extends to its ability to communicate via RF which means that there is no need to have air to send data from one sensor node to another. In the context of environmental monitoring, this type of communication proves its worth when tested underwater.

Moreover, the design of the sensor node encompasses several sensors, a microcontroller, and a ZigBee radio transceiver, namely the XBee module. Each one of these sensor nodes receives and relays sampling data back and forth from the base station. The base station is represented by the Raspberry Pi Model B, and with the help of the Xbee coordinator, stores the data received from the sensor nodes into the MySQL database. Information is sent to the database from the Xbee with the assistance of a middleman known as the gateway application and is programmed in Python. With the aid of a web interface and a web server, users can download and view the sampling data brought in by the sensor nodes and retrieved from the database server. One benefit derived from having a web interface is the ability to view sensor data updates in real-time which is very important in the context of adaptive sampling. While this design is promising and has proven to work, the coverage of the XBee radio transceiver has not been tested to its limit which can become a problem in specific environments.

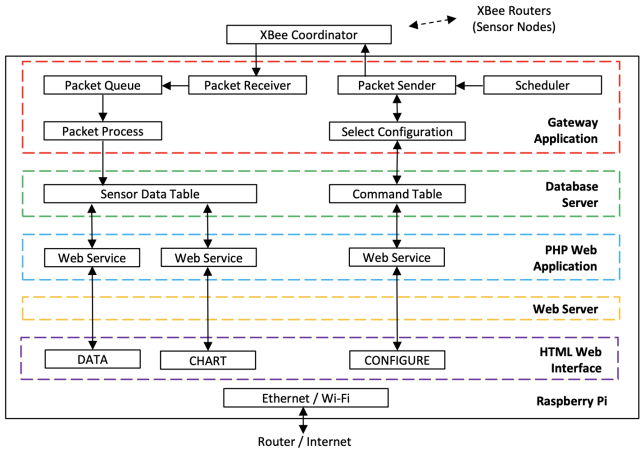

Fig. 4. Functional block diagram of base station.

Aside from the type of networks being implemented with the Raspberry Pi, it would be careless to not mention the impact of having a distinct network protocol. One highly regarded protocol that has been tested is the OLSR (Optimized Link State Routing) on a WMN testbed [12]. OSLR is run in parallel with TCP (transmission control protocol) which makes routes for data transmission such that no one route is bogged down with so much data that it cannot be relayed to the receiver. A routing table is maintained inside every node and is filled with a list of its neighboring nodes. OLSR uses HELLO messages to locate one-hop and two-hop neighbors and allows the sender to compare all the routes containing either of these types of neighbors. The sender can make a decision using the Multi-Point Relays (MPR) which connects to the neighbors thus avoiding multiple hops of data dissemination and choosing one-hops.

Although the hardware, namely the Raspberry Pi and its components on the board, are the same for our purposes, a better approach to the connectivity of all the Raspberry PIs and their communication with each other revolve around the BATMAN Advanced protocol. The CPU on the board is an ARM processor with 700 MHz clock speed. CPU performance can be compared to a Pentium II 300 MHz processor and the GPU performance is similar to the original Xbox. It has a variety of interfacing peripherals, including a USB port, HDMI port, 512MB RAM, SD Card storage and interestingly 8 GPIO port for expansion [8]. Like OLSR, this protocol is a multi-hop routing protocol used for deploying mobile ad-hoc mesh networks. Each node has access to a routing table containing all the possible hops or routes to other nodes. Unlike OLSR, this routing table is only used to communicate with the nodes

what hops are available for which they are to choose the best one possible that leads to the nearest neighbor [2]. After a decision is made as to which route they will choose to hop the packets to their nearest neighbor, the routing table is no longer maintained. This is where the brilliance lies because this protocol aims to reduce the overhead as much as possible. When the addition of nodes causes the wireless mesh topology to become wider, the link-state algorithm in the OLSR protocol is unable to account for the dynamic change in the mesh topology and update the status in real-time. BATMAN-Adv is the alternative to the HWMP (Hybrid Wireless Mesh Protocol). HWMP contains AODV or Ad-hoc On-Demand Distance Vector.

In an experiment comparing the performance evaluation of ADOV and BATMAN, three scenarios were performed such that each differed in the placement of nodes in a two-story building and the node locations [14]. For scenario A, three netbooks were placed in adjacent rooms of the building during working hours. All nodes were in direct radio range such that no routing protocol was necessary. Thus, packets could only be routed using single-hop. In scenario B, fourteen netbooks were distributed across four different rooms where two pairs were on different floors. The dashed lines show reinforced concrete walls that significantly affected radio conditions.

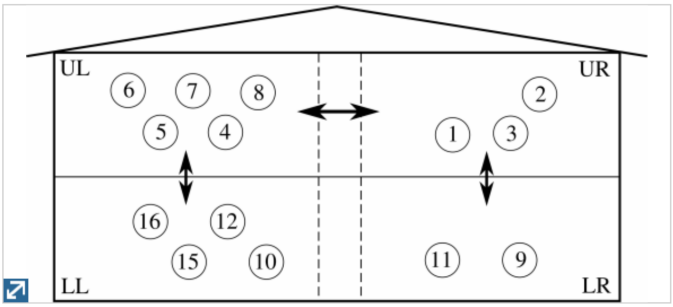

Fig. 2: Node distribution for Scenario B. The upper/lower left/right section is denoted by {U,L} {L,R}. Dashed lines show reinforced concrete walls that significantly affected radio conditions.

As shown in the figure, the use of BATMAN-adv enabled the nodes in the vertically adjacent sections and the upper-level sections to communicate with each other thus becoming successful at creating a viable network. ADOV selects the possible neighbors that can be connected with the utmost signal strength. Links that form paths for packets to flow are only unblocked when the received signal strength (RSS) for the neighbor exceeds a certain threshold. Likewise, if the node on the opposite end fulfills this requirement, the neighbor signals to it and the link that connects the two is formed to enable higher-layer communication. However, if these thresholds are not met, the link is cut off. Due to ADOV's picky neighbor selection process, only nodes in sections UR and LR could communicate with each other. In scenario C, seventeen netbooks were distributed over one level after working hours. Although the interference was low in this time frame, the results were similar to that of scenario B.

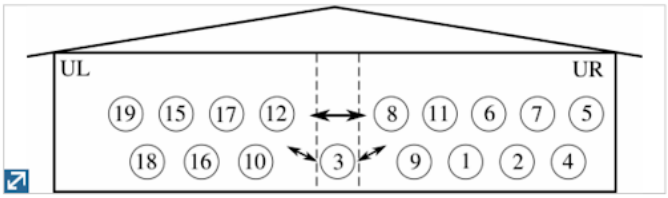

Fig. 3: Node distribution for Scenario C. The upper left/right section is denoted by U {L,R}.

In short, BATMAN proved to work significantly better than ADOV in areas with strong interference and over long distances. ADOV failed to produce stable routes in a multi-hop setting that was possible. Hence, it is more practical to use BATMAN Advanced for autonomous underwater vehicles due to its flexible nature and its preference of short and asymmetric links over short paths.

IV. Challenges

Some of the first challenges of this project are the process of adaptive sampling underwater since the environment is changing in ways that can not be determined before-hand. Since this project is so closely related to the one from the University of Girona, looking closely, they also show their problems lie in the fact that they cannot map the multiple robots due to the environment they are in. This can be quite challenging because the underwater environment is very hard to predict and using sensors and acoustic signals can provide inaccurate results [4]. This leads to the need for constant communication between the several robots, or at least a high rate of communication. The use of SLAM (Simultaneous Localization and Mapping) is used to overcome this but is not very effective when going to a bigger scale such as an ocean or maybe a long river. Hence, these AUVs have to operate on a larger scale to communicate and relay information to air support [3].

A major challenge to face with this is still the fact that water will attenuate any signal produced from underwater.Radio signals travel at little to no frequency underwater which disturbs the communication and can cause robots to break out of formation. This means that the actual location at one single time can be unknown. Acoustic waves are generally considered for underwater wireless communication since they do not absorb much in an underwater environment. Since they have a low absorption level, acoustic waves do not waste much energy in getting converted to other forms of energy in the water medium [15]. Acoustic waves can reach a point of contact through various paths, especially in the shallow path because of larger transmission distance compared to water depth. Therefore, if the path depth of the water body increases, there is more water pressure affecting the wave transmission, causing path loss for the acoustic waves and increasing noise interference [9]. These challenges of noises and path loss can highly affect the communication links, causing errors or link outages which makes it challenging to use acoustic waves in a water body with impurities, fast-moving tides, and deep waters [7].

Therefore, for this project, data-retrieval will be conducted once the vehicles read the data underwater, using their sensors, and rise back on the surface.

Also, the current of the ocean or river can carry the vehicles down the river or into the ocean which is very hard to predict.

V. Plan of Work

What needs to be addressed in the future is to first figure out how to implement batman-adv to connect the Raspberry Pis. Doing so controls each of the robots together through a tough medium that is water which would allow for a non-master-slave relationship. This means that communication between any of the Pis is not dependent on a particular Pi and would optimize sampling data from the ecosystem, specifically underwater. A problem that may arise is if there is no communication for a long period in that nothing is interesting in the environment. Hence, the Pis may start to drop from the network, but with multitasking, this can be mitigated.

Using distributive adaptive sampling aids in proper data retrieval over the samples for monitoring the environment with sensor networks. Numerous sensors provide continuous data to the main server. The communication of sampled data at every sensor in the AUVs is maintained based on the rate at which the central sensor holds the data load. For this approach, the interval between two consecutive samples must be first collected and adjusted accordingly. That interval is adjusted in the desired range based on the data-streaming requirements and attributes. If the range modifications are more than what is allowed by the sampling interval range then another range of sampling data is requested from the main server.

VI. Proposed Solution

There are two ways to go about forming a network mesh connection among all the Raspberry Pis. They are done by implementing centralized or distributed networks. The centralized network is a system in which all nodes of a communication network receives and sends packets of information via one hotspot or main router. However, these single-point central controllers are subject to possible failure. When this happens, all nodes are at a disadvantage which means that none of them can either send or receive information. For adaptive sampling purposes, it is preferred to have these packets distributed from node to node via the distribution network. Adaptive sampling relies on a multiagent coordination scheme which, in turn, can make local decisions on global information more accurate. The whole idea behind this decentralized method is to allow nodes to communicate freely to any other node any information they find in a given time. In short, the consensus theory-based multiagent coordination scheme is more suitable for decentralized control in AUVs since there is no central controller (or sink node) to collect the network-wide information and make overall decisions accordingly [11]. Our decentralized network is supplemented with some data transmission in which large files are transmitted between the host Pi and the receiver of varying size types. This is vital to coordinating different movements of all the Pis after the first phase of adaptive sampling. During the first phase, all the pis gather all sensitive information in their respective vicinity while maintaining close contact with each other. Each pi's obtained information is different than the others and when they reconvene, the acquired

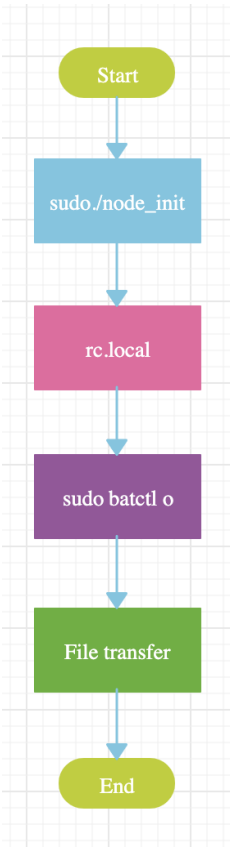

Fig. 3: Procedures for initializing the network for file transfer

data is shared between them so that during the next phase, the AUVs will not revisit the same area as last time and waste time. To enable Pi communication, python code is created for the server and client to monitor the time it took for the information to be sent as well as the byte transfer. It bridges the gap between the two Pis together allowing free communication without any reliance on a centralized router.

To initialize the multi-node connection, we installed BATMAN scripts that would run the procedure to start the connection between the Pis via the LAN. This LAN represents the bridge where information can be acquired from an external source, which is the river in our case. The startup constitutes specific shell scripts via rc.local, a system paramount to establishing a decentralized network upon boot of the raspberry pi. One specific script executed upon boot via rc.local is 'sudo batctl o'. This command pairs one node with another batman adv node in this range to look for neighbors. Once started, the addresses can be seen under the command in which one of them is listed as the originator and the other is listed under Nexthop. For multiple nodes of at least 4 or greater, there would be a list of nodes under these categories. Inserting a node into the mesh would require running `sudo./node_init` separately apart from rc.local prior to rebooting the raspberry pi.

The roles of the nodes are master and slave between a pair of nodes even if there are more than two nodes. This applies to the direct transmission of packets containing information from one node to another. For example, in one instance, node

A can send information to node B via a large file transfer. Node A would be the master and node B would be the slave. However, Node B can now be the master for another node that is one of its neighbors depending on which one of its neighbors in the single hop list has the best link quality in the neighbor table. The node with the best link quality means that it is closest to the master node due to reduced latency. Consequently, this neighbor is routed directly to the master. There are times when single hops are unavailable or would complicate the transmission of information, in turn, increasing overhead. In a distributed network, different paths will need to be taken but not by a direct line via single hop. A single hop neighbor will need to be routed via another single hop neighbor before reaching the final node. The intermediate node will be a slave at one point in time and a then a master after it receives information. In turn this new master would send packets to the final node.

When a node is added or removed from the network, the other nodes are not affected as they don't require one to transfer information to the next. One can add a node easily by calling the IP address of the router as the base station in which the computer gets its data from. A node will be dropped off once it is out of range of the router which will vary. The information that the node carried will be lost but if they are far enough out of range of the router then there is information there that is not in our area of interest at that time.

In the event of master/slave interference, multiple transmissions at one time are handled by delete the transmission. That way there there's no conflicting data. Since there are many nodes transmitting frequently multiple transmissions will not affect the overall gain of the testing. If there is heavy interference on one of the nodes then it isn't taken into account when adaptive sampling is taking place such as obscure data or incomplete information.

VII. Performance Evaluation

To connect two Raspberry Pis, the batman-adv commands were run on the Pis' terminals to set up the communication between the two by connecting them on a custom mesh network, named "my-mesh-network". This is the network on which the Raspberry Pis are connected to establish the ad-Hoc connection. The mesh connection is verified with some of the multiple network characteristics. The bat0 and wlan0 links verify that the mesh network is established between the Pis by sending and receiving some test packets. An address under the Originator indicates the available Pis for communication under the given network. After setting up the two Pis together, communication was tested by adding a server and client Python files on the two Pis, configuring with each Pi's IP address. Once the files were added, calling the files on the Python level of the terminal verified that the files were sent an received on the two IP addresses of the Pis.

After the connection was set up between the two Pis, a mesh-bridge network was established to incorporate connections for more than two pis that communicate to the main router from the bridge. The main idea with setting up the mesh-bridge is to ensure connectivity with the router, which

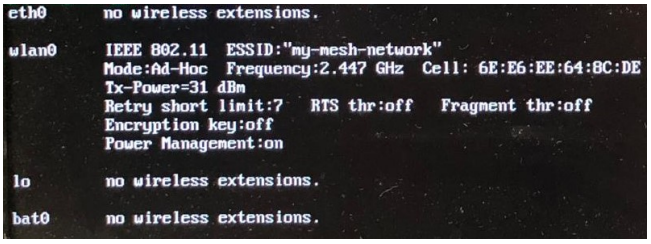

Fig. 4: The image above shows the network has been made via "my-mesh-network". This is important to verify as it is how the pis communicate

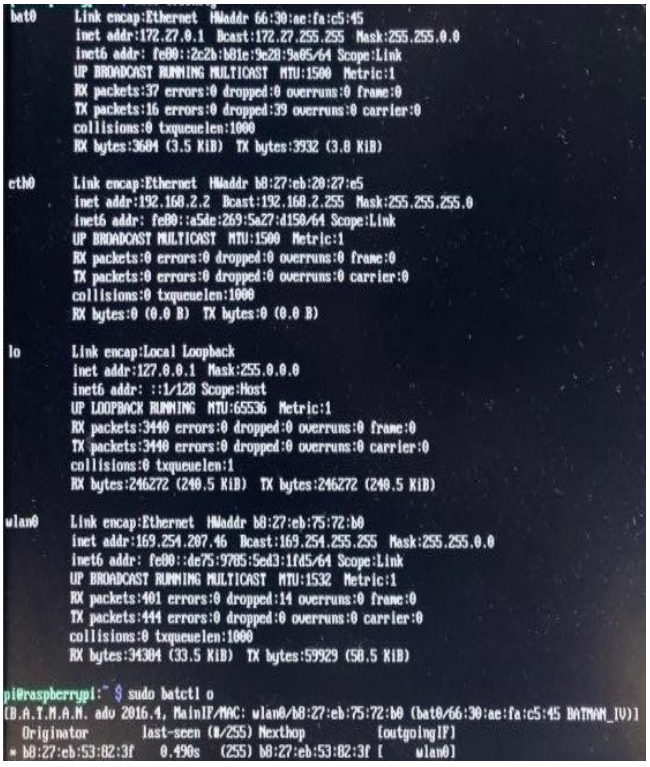

Fig. 5: This figure shows that the batman connection has been established and everything is ready for communication. This is shown with the bat0 above

is a non-batman-adv device. Once the mesh-bridge network was established, a third Pi(again shown under the Originator column) was introduced and the communication was verified by sending and receiving packets across the three devices, by verifying if a certain IP of node was under the network. The batman-adv commands were flashed in the Pi memories in their startup scripts so that the network connection is setup between all the powered Pis as they power on. Connection is only maintained for the Pis that are on. Therefore, shutting down a Pi simply removes it from the connection and the network is maintained with the remaining Pis.

To measure the reliability of the data transmission on a batman connection, multiple files of increasing size are sent and received between two Pis. The time proportionality is measured as each file's data is read until the end. There were three random files created of size 10mb, 30mb, and 60mb.

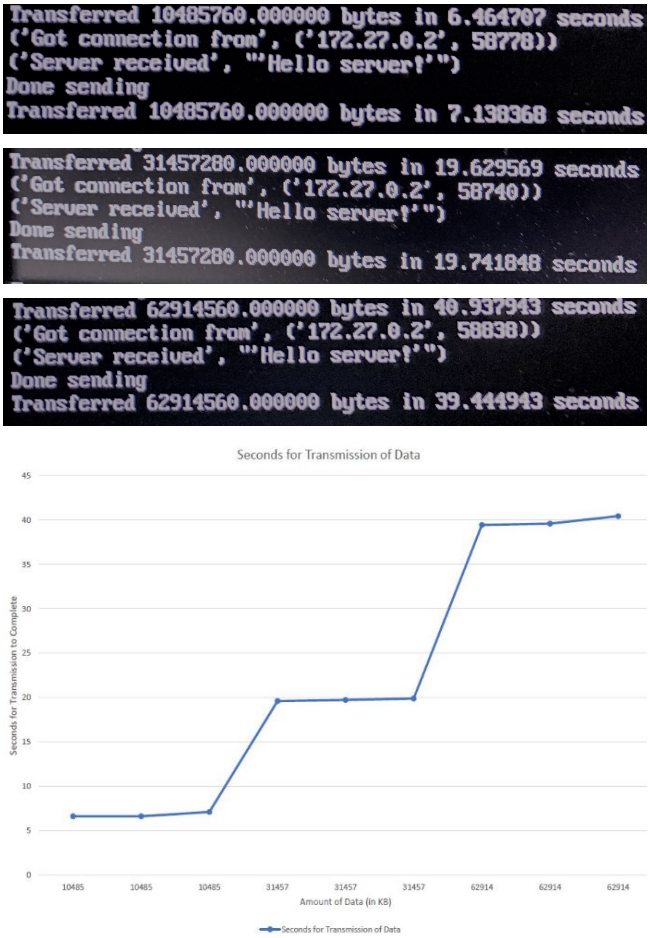

Fig. 6: These figures show the speed of connection when sending messages between nodes , which includes sizes and the amount of time it took, representing linear relationship. This was taken in the lab under controlled conditions.

Each file was sent and receiver over TCP protocols to measure the time it took for data transmission for several test runs. On average the 10mb file took 7.13 seconds to transmit its data, 30 MB file took 19.7 seconds, and 60 MB file took 39.4 seconds. Approximately, the time of data transmission was proportionate to the size of the file. Hence the Pis were made capable of sending large files reliably in linear time-fashion through the mesh connection of batman-adv. This reliability can further be tested by configuring three or more pis, where each Pi can be connected to varying sensors and transmit those data to each other and coordinate movements.

For communicating the sensor data, as a test run, an oxidation-level measuring sensor is connected to one of the Raspberry Pis. For this experiment, the machines performing adaptive sampling take decisions based on couple of sensors attached, measuring oxidation levels and temperatures. As the sensor reads the data, it saves a large amount of data in a CSV file and stores it on the Pis memory. Then, the Pi sends the file to connected Pi by reading the data and writing on its memory. As the Pi is done receiving the file, it triggers an LED(which was connected along with the sensor) to light up and indicate movement coordination. In the actual implementation, the lighting of LED is substituted with the vehicle making a specific movement, ideally in response to the type of data received. The file transfer protocols took place in increasing order of file sizes based on the amount of sensor data chosen to write in the CSV file to further ensure the reliability of data transmission. Upon the performance, all the data was accurately stored and transferred without any loss or errors.

After simulating the conditions with the LED bulb, the next goal was to implement sensor function responses on the Pixhawk vehicle. The scripts that trigger different movement channels for the vehicle were loaded and test with some channel triggers. Upon successful trigger responses, basic adaptive sampling conditions were developed, in which the sensors on the Pixhawk vehicle read the oxidation level values for a fixed amount of time and stored the values in a list. After storing the values in the list, the script performed arithmetic functions to calculate the average of the values stored in the list. Once the average was outputted, the script triggered the vehicle to sense the value one time and compare the value with the average calculated. If the value sensed was greater than the average value, then the vehicle runs its forward channel or else reverse channel. This type of rudimentary setup was to show the ability of the Pixhawk to move a certain upon finding desired data. Once that was established, the Pixhawk vehicle was programmed to have abilities to run in a path and sense data simultaneously, making the vehicle ready for pool testing. To get the vehicle to sense the data while movie in a path, the script is fed with threading functions so that the vehicle moving and data sensing functions got joined.

Before the pool testing, the vehicles were fed with adaptive sampling algorithms, built off on a reward system and moving the vehicle in a grid. For this, the algorithm triggers the vehicle to sense and record data of the chosen parameters, temperature and oxidation levels for this project, for a starting grid point. Based on the data read in the grid, the algorithm calculates the variance of the data, labels reward points to triangular sub-regions in the grid. The algorithm triggers the vehicle to move in the direction of the highest reward, indicating the region of interest and repeats this process until it locates the grid with the highest reward(indicated with the big blue circle).

Upon conducting the pool tests, each of the two vehicles released in the pool tested one of the two parameters and the communicated data. The vehicles ultimately reached a region with the lowest temperature and a specific oxygen level within the pool domain, indicating the successful implementation of adaptive sampling.

VIII. Conclusion and Future Work

This project showed the importance of having adaptive sampling and how the nodes can communicate in a way that adaptive sampling benefits from. The results shown above indicate that the use of an ad-hoc network in a way that there is no leader node is faster and more efficient than the other previously used networks. Next, adaptive sampling as a whole proves to be more efficient at scanning an area for areas of interest than the other used methods of lawnmower style observation.

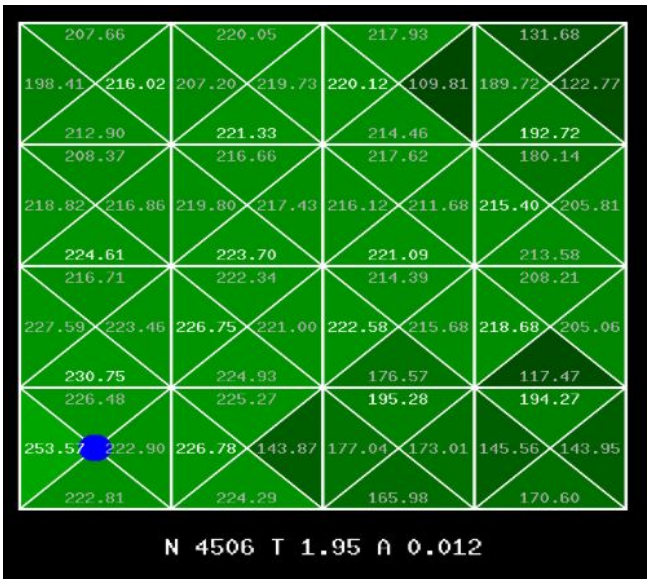

Fig. 7: Adaptive sampling simulation to represent the reward implementation during the pool testing

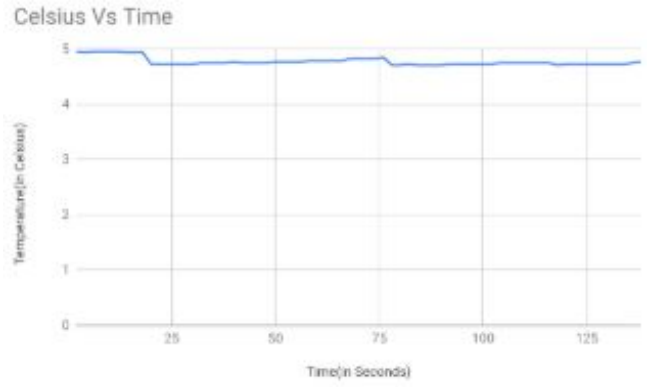

Fig. 8: Temperature of the water in the Raritan river taken over a period of time to show the slowly changing amount

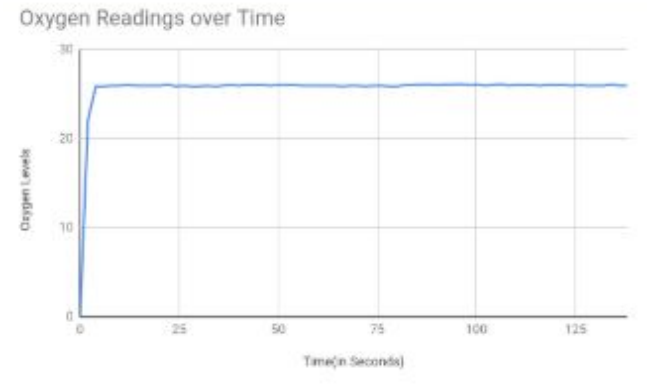

Fig. 9: Oxygen readings taken in the Raritan river to show that there is little to no change even with adaptive sampling where we were taking measurements

There can be an expansion of this project into using more nodes and even combining more environments such as land and sea or sea and air such that the data is consolidated leading to more accurate predictions. Another possible application of this project is to upload data about a creature of interest and have the vehicles scan an area in the environment to search for this creature, such as an endangered species. This would prove to be a better use of one's time as for now, humans are doing this searching with little technology usage.

REFERENCES

[1] Edoardo M Airoldi and Kathleen M Carley. Sampling algorithms for pure network topologies: a study on the stability and the separability of metric embeddings. *ACM SIGKDD Explorations Newsletter*, 7(2):13–22, 2005.

[2] Nuzli Mohamad Anas, Faiz Kamil Hashim, Hafizal Mohamad, Mohd Hafiz Baharudin, and Mohd Pazli Sulong. Performance analysis of outdoor wireless mesh network using batman advanced. In *2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 1–4. IEEE, 2015.

[3] Ricard Campos, Nuno Gracias, Albert Palomer, and Pere Ridao. Global alignment of a multiple-robot photomosaic using opto-acoustic constraints. *IFAC-PapersOnLine*, 48(2):20–25, 2015.

[4] Ricard Campos, Nuno Gracias, and Pere Ridao. Underwater multi-vehicle trajectory alignment and mapping using acoustic and optical constraints. *Sensors*, 16(3):387, 2016.

[5] Sheikh Ferdoush and Xinrong Li. Wireless sensor network system design using raspberry pi and arduino for environmental monitoring applications. *Procedia Computer Science*, 34:103 – 110, 2014. The 9th International Conference on Future Networks and Communications (FNC'14)/The 11th International Conference on Mobile Systems and Pervasive Computing (MobiSPC'14)/Affiliated Workshops.

[6] Ankur Jain and Edward Y Chang. Adaptive sampling for sensor networks. In *Proceeedings of the 1st international workshop on Data management for sensor networks: in conjunction with VLDB 2004*, pages 10–16. ACM, 2004.

[7] Arijit Khan and Lawrence Jenkins. Undersea wireless sensor network for ocean pollution prevention. pages 2 – 8, 02 2008.

[8] Donald Knuth. Knuth: Computers and typesetting.

[9] Liu Lanbo, Zhou Shengli, and Cui Jun-Hong. Prospects and problems of wireless communication for underwater sensor networks. *Wireless Communications and Mobile Computing*, 8(8):977–994, 10 2008.

[10] Paul J Lavrakas. *Encyclopedia of survey research methods*. Sage Publications, 2008.

[11] Hao Liang, Bong Jun Choi, Weihua Zhuang, Xuemin Shen, ASA Awad, and Atef Abdr. Multiagent coordination in microgrids via wireless networks. *IEEE Wireless Communications*, 19(3):14–22, 2012.

[12] Tetsuya Oda, Masafumi Yamada, Ryoichiro Obukata, Leonard Barolli, Isaac Woungang, and Makoto Takizawa. Experimental results of a raspberry pi based wireless mesh network testbed considering tcp and los scenario. In *2016 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 175–179. IEEE, 2016.

[13] Mark R Patterson and James H Sias. Modular autonomous underwater vehicle system, November 30 1999. US Patent 5,995,882.

[14] Daniel Seither, André König, and Matthias Hollick. Routing performance of wireless mesh networks: A practical evaluation of batman advanced. In *2011 IEEE 36th Conference on Local Computer Networks*, pages 897–904. IEEE, 2011.

[15] Zhi Sun and Ian F Akyildiz. Magnetic induction communications for wireless underground sensor networks. *IEEE Transactions on Antennas and Propagation*, 58(7):2426–2435, 2010.