Autonomic Cooperation Strategies for Robot Swarms

Catherine Saunders
Supervised by: Roy Sterritt, George Wilkie
School of Computing and Mathematics
Ulster University
Northern Ireland, UK
E-mail: saunders-c2@email.ulster.ac.uk, r.sterritt@ulster.ac.uk, fg.wilkie@ulster.ac.uk

Abstract— **In this paper, we describe two strategies that allow a swarm of simulated robots to cooperate. For a swarm of robots to function cooperatively, self-management and autonomy are essential. Direct communication is used to enable swarm entities to communicate. The research aim is to evaluate various architectures and protocols for cooperation strategies that enable swarm robots to ask for, and respond to requests for assistance. The work is in two phases. Only phase 1 is described in this paper. The first phase involves the creation of simulation environments of robot swarms. This phase enables us to develop, evaluate and refine suitable architectures and protocols for swarm cooperation. Using simulation, it is possible to assess swarm cooperation in the large (essentially hundreds or thousands of robots per swarm). In the second phase, the cooperation protocols developed from phase 1 will be trialed on a small number of physical robots, to evaluate the complexity introduced from the real world. The 1st architecture simulated features a hierarchy with Ruler robots communicating with a swarm; the Ruler robots can request the help of the swarm. The swarm is only able to respond to the Rulers, intra-swarm communication is not possible. In the 2nd architecture simulated, a non-hierarchical homogenous swarm is able to communicate by posting help messages to a centralized Message Board entity. In future experiments, architectures involving the incorporation of a Message Board role within each swarm robot, thus removing the disadvantages associated with having a centralized component, will be explored.**

Keywords- Robot; Autonomic Computing; Swarm; Simulation; Cooperation;

I. INTRODUCTION

The purpose of this research is to investigate cooperation strategies that will enable a swarm of robots to collaborate and perform a task without human involvement. The aim is to show how direct communication could work within a swarm scenario. Most swarm research tends to focus on indirect communication, this normally involves changing the environment to influence other swarm members, or responding to other swarm members in order to maintain distance and mimic flocking. In future, we want to compare the performance of different approaches and determine the optimal cooperation strategy for swarm collaboration. We are interested in direct communication in the form of messages sent via a central controller or by direct communication from robot to robot. We are working on creating simulations to test various cooperation strategies; this paper will focus on our current research and conclude with future research ideas.

Autonomic Computing [1] takes its name from the Autonomic Nervous system, which can maintain bodily functions independent of conscious thought [2]. The aim of Autonomic Computing is to improve the self-management of autonomous software systems. This paper is part of a research project that seeks to design a model, which will allow swarm entities to communicate information in order to collaborate as a whole. In order to do this, each entity in addition to being self-managing, must be capable of receiving and reacting to communications from other swarm members. Within Autonomic Computing, there is the concept of an Autonomic Element (AE), which consists of an Autonomic Manager (AM) and a Managed Component (MC). Having an AM that uses a feedback loop to constantly check on the state of a system is particularly applicable to a robot swarm [1]. The hardware of each swarm robot represents the MC, the AM is the software that must monitor the battery life, component state, and direct the local behavior of the robot. Robot AM's must be able to cooperate with each other in order for the autonomic swarm to function efficiently. To be truly autonomic, a system must be Self-Aware, Environment-Aware, Self-Adjusting and Self-Monitoring [3].

Space exploration is an area that could benefit from incorporating Autonomic Computing ideas. Future space missions will seek to go beyond the monolithic rover concept and instead feature multiple autonomous rovers or spacecraft. It is important that Autonomic self-management techniques are incorporated into future missions that feature multiple entities. It would not be feasible for humans to manage the actions of every member of a swarm, especially in an emergency situation. NASA's concept missions demonstrate their interest in more ambitious fully autonomous swarm exploration.

The NASA Autonomous Nano Technology Swarm (ANTS) project features a concept mission known as the Prospecting Asteroid Mission (PAM). This mission would involve sending 1000 spacecraft to explore an asteroid belt. The reason for sending a large number of spacecraft is to counter the expected large-scale decimation of the swarm [4]. The mission would include 10 scientific instruments, with each spacecraft carrying only 1 instrument; we are using this example as inspiration for our research scenario. The swarm would feature different roles, such as Ruler, Messenger and Worker. Autonomic computing ideas are essential in order to ensure that swarm entities are self-managing and able to cooperate effectively with each other [5][6].

In this paper, we describe two simulations that feature a swarm of robots using direct communication in order to cooperate. Section II gives an overview of related work and the different approaches within swarm research. Section III describes the two different approaches and the C# simulations. Future work is discussed in Section IV; this will involve further simulation work in order to decrease the number of unanswered help requests, and also testing on mobile robots

II. Related Work

Within swarm research there are 3 general types of systems that have been explored; centralized, decentralized and hybrid approaches. A centralized system consists of a central controller that collates data from swarm members; this enables it to intelligently co-ordinate how each swarm member should behave [7]. The disadvantages of this type of system are that it does not scale well, the larger the swarm, the less efficient the controller will be at processing information and coordinating the actions of the swarm. As there is a central controller, any damage incurred can negatively affect the behavior and performance of the swarm and may also jeopardize the overall mission [7].

In contrast to this, a decentralized system operates in a Peer-to-Peer manner with communication occurring between swarm members. A Peer-to-Peer approach helps to avoid the bottleneck that can occur when there is a central controller processing all swarm communication traffic. Another advantage of Peer-to-Peer approaches is that if swarm members are damaged, the swarm can still function, as no member is indispensable. In a centralized system, if the central controller becomes damaged, the swarm would no longer be able to function cohesively [7].

A hybrid system results from the combination of centralized and decentralized strategies to varying degrees. A hybrid system can take the form of a decentralized swarm were the communication takes place locally but also includes another supervisory element that analyzes global data and provides overall mission direction [7]. The NASA PAM swarm fits the definition of a hybrid model as it features a hierarchy of Rulers, Messengers and Workers. The Rulers would be able to coordinate the behavior of the Workers by organizing them into teams and choosing exploration targets [4].

A decentralized Peer-to-Peer model is presented in [8], where a navigation system is proposed with each robot within a swarm maintaining a table of location information of every other robot within the swarm. The robots broadcast messages with location information to their neighbors; this is then distributed throughout the swarm. To test the messaging protocol, a robot declares itself to be a target, other robots must move towards this robot using the location information they have received and stored in their table. The authors were able to test their research on physical foot-bot robots [8]. This is something we would like to do in future. Another example of broadcasting within a swarm is explored in [9], the authors created a Global Coordination System, where information is exchanged by agents within a decentralized swarm by using a wireless sensor network. Each robot starts from the same position within an indoor environment, they are able to keep track of their position by checking how far they have moved from their start position. To help the swarm search for targets, robots use the location information being broadcast by other swarm members.

There is not as much research dedicated to decentralized direct communication between members of a robot swarm. Most swarm research tends to focus on implementing a system that is either centralized or one that is decentralized and uses indirect communication techniques, such as pheromones. Indirect communication can be achieved by changing the environment in a way that influences the others that are operating in that environment. This is known as Stigmergy; it is seen in nature and is the basis of much bio-inspired research. A virtual pheromone approach has been explored in [10], robots are placed at random positions within an arena and given the task of sweeping the perimeter. A pheromone trail is left by each robot to notify others that an area has already been mapped.

For the NASA PAM mission and other future missions that involve multiple robots carrying out complex tasks, it may be necessary to equip robots with more intelligence than is present in a purely reaction based system. In [11] an interesting hybrid approach is discussed, a cluster of 3 robots begin their mission as a swarm but can change to a master/slave configuration when cooperation is required. The robots are set the task of finding an object; if a robot finds the object it assumes the role of Leader and sends the location to the other 2 robots. The slave robots cannot communicate with each other, only with the Leader, the Leader then broadcasts the data collected from both slaves.

Role switching is also featured in the SWITCH project [12], which was developed for the RoboCup competition [13]. The RoboCup is held annually and has robots competing in teams to play soccer; the aim of the competition is to improve many areas of computer science including cooperation between robots. The SWITCH robots are able to change their role from Striker to Defender in response to how the game is progressing; each role has its own goals and strategies. This idea of being able to change roles depending on the situation could prove useful, as there are benefits to both the centralized and decentralized approaches. Being able to switch between the two is a useful adaptive technique that would allow a swarm to operate under centralized control but without the disadvantages this brings.

We have not implemented roles in our current simulation research but may consider it in future as a useful failsafe mechanism. Specifically, if a swarm is routing messages via a centralized communication element, which becomes damaged, the swarm may no longer be able to cooperate. By using roles, any member of the swarm can self-nominate and become a central controller, thus the limitations of a centralized system are avoided. The self-nominated robot would cease their exploration task and change their 'role' to that of a centralized communication element.

III. CURRENT WORK

This section describes two simulations that we have designed and implemented in C#. The systems are a mixture of both decentralized and centralized approaches, in both, each swarm robot is autonomous. We use a central Message Coordinator in the 2nd simulation, which could fall prey to the dangers of a centralized system. In future we would like to incorporate the coordinator as a role into each swarm robot. A simulation is useful as it enables us to create numerous robots without being constrained by the limitations of physical hardware. To ensure rigor, it is also necessary to perform real life experiments, since unexpected results can arise from the complexity posed in the real world. The final stage of our experiments will involve testing our cooperation models using a small cluster of four Dr Robot X80-H robots [14].

In Section III-A, we describe a system that features a Ruler AE, which can communicate with every member of the swarm. Sending a message to every swarm robot AM each time help is required is not very efficient. It is simple to accomplish in a simulation but could prove more challenging with actual robots. As a result of this, we created another design with a dedicated centralized Message Board. The simulation described in Section III-B features a homogenous swarm as opposed to the hierarchical system described in section III-A. Swarm robots still cannot communicate with each other and must use a central Message Board to post Help requests. When a robot finishes its task it checks the Message Board for help requests that it could fulfill.

A. Implementation 1 – Rulers and Workers Hierarchy

Most swarm research seeks to mimic natural systems e.g., flocking, shoals, foraging; however, we are interested in direct communication and instilling more intelligence in each individual swarm robot. In this simulation, our scenario involves a robot finding an interesting feature that requires other members of the swarm to move towards its location and assist. This version was inspired by the NASA PAM project; it features 3 Ruler robots that communicate directly with a swarm. The scenario involves Ruler robots requesting help from members of a swarm when they encounter interesting terrain that the Ruler cannot traverse. Swarm robots only respond if they have not reached their target destination and begun their own experiments. When the Rulers discover an interesting feature situated within terrain that they cannot traverse, they send a message to the swarm asking for help. The Rulers can communicate with every member of the swarm but the swarm can only communicate with the Ruler robots and not with each other. The swarm only replies to help requests and cannot initiate contact with the Ruler.

The map in Figure 1 includes orange and black tiles, which represent interesting terrain features that the 3 Ruler robots encounter during the course of the simulation. The user specifies the number of robots in the purple swarm. Each robot runs in its own thread, its start position, terrain capability and mission priority are all randomly generated. The help request that is sent includes the Ruler's mission priority, location and terrain encountered (i.e., Terrain capability required). This is then used by each swarm robot to decide whether or not to respond. If a swarm robot's mission priority is higher than that of the Ruler, they ignore the help request. There are two conditions that must be met in order for a swarm robot to respond to a Ruler's help request, they must be able to cross that type of terrain specified and have a lower mission priority than the Ruler. Those that respond to the Ruler's help request are placed on a 'Helpers List', provided they are within a certain distance of the Ruler robot. In Figure 1, the green and blue Rulers are flashing a proximity beacon; only responding robots within this beacon area are added to the 'Final Helpers List'. The Rulers then send a confirmation message to each robot on the 'Final Helper List'. The swarm robots that receive the confirmation message only move towards the Ruler robots location if they have not reached their destination and begun experiments.

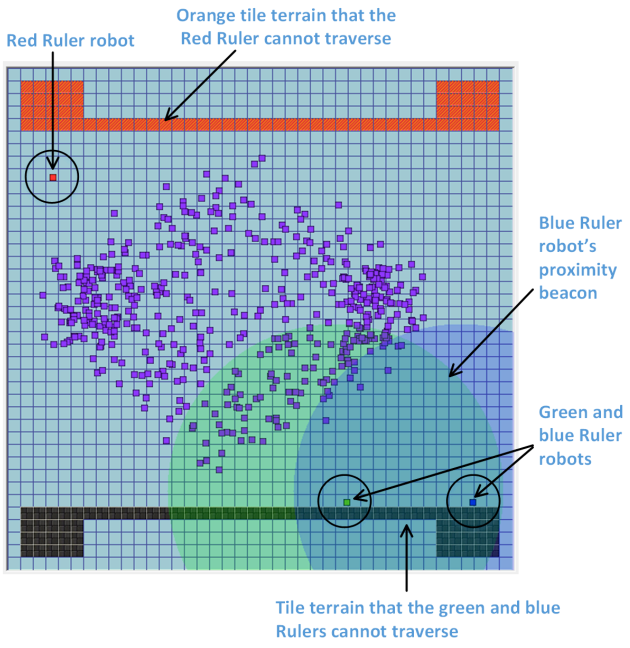
Red Ruler robot
Orange tile terrain that the Red Ruler cannot traverse
Blue Ruler robot's proximity beacon
Green and blue Ruler robots
Tile terrain that the green and blue Rulers cannot traverse

Figure 1. Ruler robots communicating with a swarm, a tile map was used with the orange and black sections representing areas that the Ruler robots could not traverse.

The flow diagrams in Figure 2 and Figure 3 show the processes (or protocols) of a Ruler robot and a swarm robot. The Ruler robot AM differs from the swarm robot AM, yet each is similar in that they both contain a continuous self-checking loop. In Figure 2, the Ruler robot creates a Helper List with the information of each swarm robot that responded to the help request. It then iterates through this list and places robots that are located within the beacon area on to the 'Final Helper List'.

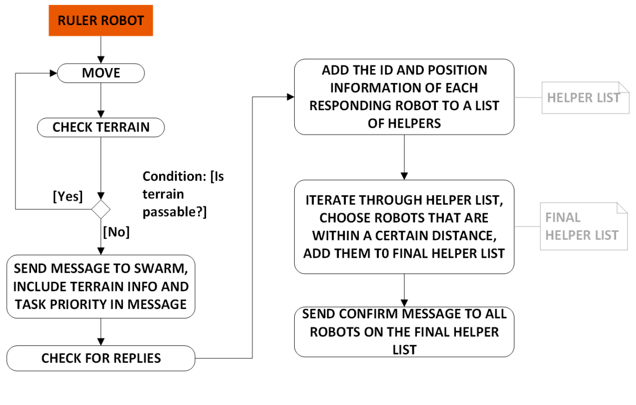
RULER ROBOT
MOVE
CHECK TERRAIN
Condition: [Is terrain passable?]
[Yes]
[No]
SEND MESSAGE TO SWARM, INCLUDE TERRAIN INFO AND TASK PRIORITY IN MESSAGE
CHECK FOR REPLIES
ADD THE ID AND POSITION INFORMATION OF EACH RESPONDING ROBOT TO A LIST OF HELPERS
HELPER LIST
ITERATE THROUGH HELPER LIST, CHOOSE ROBOTS THAT ARE WITHIN A CERTAIN DISTANCE, ADD THEM TO FINAL HELPER LIST
FINAL HELPER LIST
SEND CONFIRM MESSAGE TO ALL ROBOTS ON THE FINAL HELPER LIST

Figure 2. Ruler robot protocol

The swarm robots have an internal self-managing loop that checks for messages and formulates responses based on mission data. If the message that is received from a Ruler states a mission priority that is less important than the swarm robot's own mission priority, it will choose to ignore the message. In this respect, the swarm robots do possess some personal autonomy within the system. They may determine that their task is more important and that by refusing to help they will ultimately be benefitting the swarm mission objectives.

In Figure 3, the flow diagram (protocol) shows the loop used by each swarm robot, they only respond to messages, which have a higher mission priority than their own. If they respond to a message and do not receive confirmation from the Ruler, they continue to move to their original target destination.

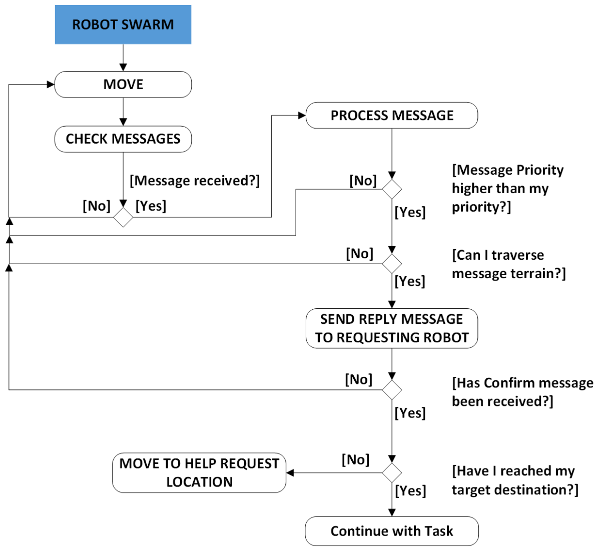
ROBOT SWARM
MOVE
CHECK MESSAGES
[Message received?]
[No]
[Yes]
PROCESS MESSAGE
[No]
[Message Priority higher than my priority?]
[Yes]
[No]
[Can I traverse message terrain?]
[Yes]
SEND REPLY MESSAGE TO REQUESTING ROBOT
[No]
[Has Confirm message been received?]
[Yes]
MOVE TO HELP REQUEST LOCATION
[No]
[Have I reached my target destination?]
[Yes]
Continue with Task

Figure 3. Swarm robot flow protocol

The system as a whole is self-optimizing, it can adapt and change its behaviour based on the terrain encountered. Rulers act as Autonomic Managers of the swarm component, they can utilize this component when necessary. The Ruler robot has the capability to reconfigure the swarm component by drawing their resources when an interesting feature is discovered. Ruler robots can reconfigure the swarm behaviour by sending a help message and choosing helpers, the system is therefore capable of self-optimizing when certain situations arise. The Ruler robots only act as Rulers in certain situations as having a swarm explore autonomously improves the chances of finding interesting features. We are using impassable terrain as a problem that necessitates help from the swarm, but this is merely a scenario to facilitate collaboration. Future work may look at using a foraging task instead where help is needed from other swarm robots to successfully forage an interesting item.

In this implementation there are 3 designated Rulers. However, in future we would like to design the swarm so that anyone can self-configure and become a Ruler when they find interesting terrain or features. Being able to change and enhance one's capability dynamically would be more autonomic and fulfill the self-configuring aspect of the Self-CHOP paradigm.

B. *Implementation 2 – Message Board and Swarm*

In this version, there are no Ruler robots, the swarm entities co-operate by posting their help requests to a Message Board AE. The Message Board is not a centralized controller; it is a passive tool that is used by the swarm to coordinate tasks, it does not provide global task direction. The Message Board is a different type of AE in comparison to the Robot AE. Within the simulation it does not exist as a visible entity, the AM is tasked with storing help requests.

In a real life experiment, the Message Board may exist as a satellite that can communicate with all robots. The Message Board could also be a swarm robot that is static and dedicates all of its power and resources to enabling communication within the swarm. The assumption being that it would be less taxing for a swarm robot to send a message to a designated static swarm robot than every other robot in the swarm. A failsafe mechanism would be to allow any swarm robot to take on the role of the Message Board. If the static Message Board swarm robot was damaged or destroyed, another robot could then nominate itself and change its 'role' from 'Explorer' to 'Message Board'. This would be more autonomic and allow the system to suffer significant losses yet still function.

The Map in Figure 4 mostly consists of green tiles that represent normal traversable terrain. The blue water tiles are used to represent an interesting feature that needs to be explored by the swarm. The simulation shown in Figure 6 features a swarm of red and yellow robots, each running in their own Thread and with randomly generated capabilities. Each swarm robot is given either a Red or Yellow color to represent a different scientific instrument; this is inspired by the NASA PAM mission, which features up to 10 instruments. In future we would like to add more instruments/colors to enrich the scenario. The start and target coordinates are also randomized, as is the speed at which each robot moves. The speed is used to work out how much battery power the robot has used; each robot starts the simulation with 100% battery.

If a robot reaches the water tiles, it sends a help message to the Message Board with details of its location, the battery life required to complete exploration, and the type of scientific instrument required. There are two instruments, red and yellow; if the robot sending the help request is a red robot, then it would need the help of a yellow robot. In the simulation, a yellow robot will always request help from a robot with a red instrument, and vice versa. The message includes the Requesting robot's ID, Instrument required, and Battery Life needed to complete the exploration task. The Message Board entity receives and stores all help messages from the swarm.

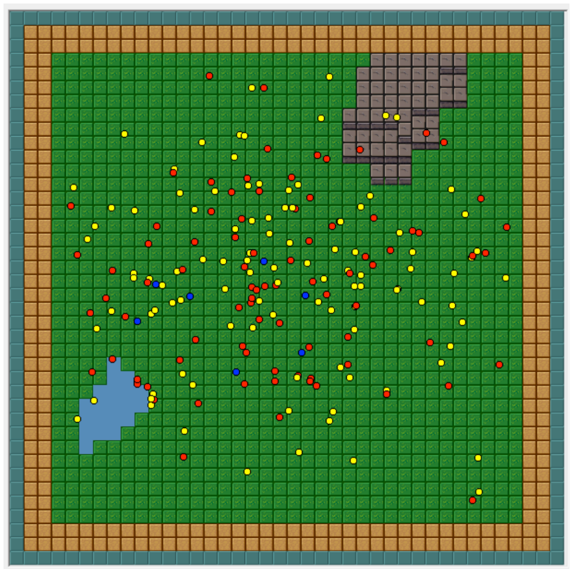

Figure 4. Swarm and Message Board simulation with terrain map. Red and yellow robots represent different instruments. Blue robots are those that are responding to help requests.

When a robot reaches its target location and finishes its task, it enters an idle state; it then asks the Message Board for a tailored list of all unfulfilled help requests. With this request, it includes its current location and instrument type. The Message Board checks the location coordinates provided by the idle robot and filters out help requests from robots outside a certain distance. It also removes any requests that do not require the idle robot's instrument type, the tailored list is then sent to the idle robot. The idle robot then chooses the help request that requires the least amount of battery power. This has led to a number of help requests not being fulfilled due to the battery life required being higher than the battery capacity of any available idle robot. Future research will look at choosing a request with the shortest time to completion window as this will help avoid a selfish swarm scenario.

When an idle robot chooses a help request, it sends a message to the Message Board and waits for confirmation. The Message Board checks that the help request is still available and unanswered, if it is, then it marks the help request as 'Completed' and sends a confirmation message to the idle robot. The idle robot then changes its color to blue and moves to the location in the help request. Unlike Version 1 of the simulation, where many robots could fulfill a help request, in this version, only one robot can respond to a help request. This is a feature that could be changed so that the requesting robot specifies in the help request how many helpers it needs for a given task.

The flow diagram in Figure 5 shows the decision-making processes performed by an idle robot. When a robot becomes idle, it asks the Message Board entity for a list of current unfulfilled help requests. The idle robot pauses until it receives a response from the Message Board, the response is nearly immediate and the wait time does not negatively impact the swarm behaviour. However if a swarm is very large, a single Message Board element might experience lag when trying to process list requests from the swarm. A bottleneck could occur and result in a large number of idle robots waiting for responses from an overtaxed Message Board. Future work will address whether it is necessary to have multiple Message Board nodes if the swarm is very large. The Message Board flow diagram in Figure 6 shows the processes followed when a Help Request is received, and also when an idle robot requests a tailored Help Request list.

The simulation requires further work in order to reduce the number of help requests that go unanswered. Currently, robots are able to choose the help request that requires the least amount of effort. Future work will incorporate a time to completion window and instruct swarm robots to respond to the request with the least time remaining.

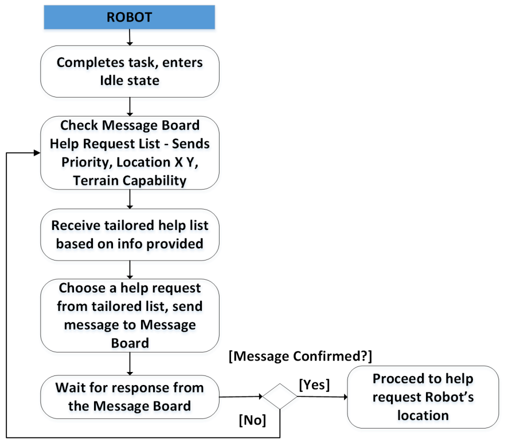
ROBOT
Completes task, enters Idle state
Check Message Board Help Request List - Sends Priority, Location X Y, Terrain Capability
Receive tailored help list based on info provided
Choose a help request from tailored list, send message to Message Board
Wait for response from the Message Board
[Message Confirmed?]
[Yes]
[No]
Proceed to help request Robot's location

Figure 5. Robot decision-making flow diagram

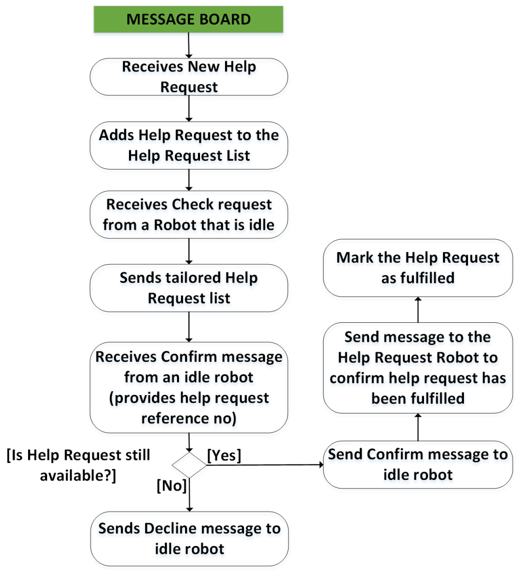
MESSAGE BOARD
Receives New Help Request
Adds Help Request to the Help Request List
Receives Check request from a Robot that is idle
Sends tailored Help Request list
Receives Confirm message from an idle robot (provides help request reference no)
[Is Help Request still available?]
[Yes]
[No]
Sends Decline message to idle robot
Send Confirm message to idle robot
Send message to the Help Request Robot to confirm help request has been fulfilled
Mark the Help Request as fulfilled

Figure 6. Message Board responding to a help request

IV. Future Work

Section IV-A describes features that could be added to the simulation in order to improve the cooperation between swarm entities. Section IV-B discusses the final stage of our research, which will involve testing the cooperation strategies on physical mobile robots.

A. Simulation Experiments

Our current approach involves a swarm of robots and a Message Board entity, however if the Message Board is damaged, the swarm has no way of communicating. Another approach would be to design a Message Board coordinator role that any robot can switch to. The system would then benefit from the advantages of a centralized and decentralized design.

In order to quantify the performance of the different strategies, future work will look at using repeatable randomized data that can be uploaded into the simulation. We would like to see whether having the central Message Board tailor the list or letting the idle robot do this locally makes a difference to the performance, i.e., the number of help requests that are left unanswered. We also plan to enrich the map terrain and add more scientific instruments/colors (capabilities) to the swarm mix.

Another feature we are interested in implementing involves stopping a robot that is in the process of responding to a help request. This would happen if a robot that is closer to the help request location becomes idle and available. If a robot becomes idle and is closer to a help request location than a robot that is currently en route, the idle robot could take over the task. This would involve sending the idle robot a list that includes help requests that are currently being answered as well as those that are unanswered.

To make the simulation more realistic, we plan to modify the Help Request format to include a time to completion value. This is an estimate created by a robot that has encountered an object or feature that needs to be investigated within a certain time frame due to suspected perishability. In addition, it may be useful to equip the Message Board AM with the ability to detect help requests that have not been answered and where their time to completion is running out.

This paper describes a work in progress, future publications will detail the results of the strategies and compare performance. The goal of the research is to determine if certain collaboration strategies are more suitable to certain situations. We want to measure efficiency of each strategy, metrics will therefore be recorded, these may include: number of steps taken by each swarm robot, number of help requests made, number of help requests that go unanswered. The time taken by each strategy is also important if a foraging scenario is adopted, the time taken to find all items can be compared.

We are currently working on implementing both collaboration strategies into a new simulation where the terrain is identical; this will help us gather statistics on how well they perform in relation to each other. In addition to these strategies we are also implementing a local broadcasting collaboration strategy, which uses nearest neighbour message passing. In a real life scenario this would help solve any signal range communication problems.

Another aspect of our research is to include an Autonomic Overseer Element that can monitor the simulation as it is running, assess the metrics being recorded and then instruct the swarm to change the collaboration strategy it is currently using to one that the Overseer deems more suitable to the terrain or situation. A terrain that is hazardous could result in more swarm casualties, in this sort of situation using a decentralized communication technique that requires a message to be passed via close neighbours (local broadcasting) may be unsuitable. If the swarm is spread too thin then the messages may never reach a sizeable number of robots. The Autonomic Overseer would conclude that the swarm is operating inefficiently and change their mode of communication to a centralized communication strategy. It may choose the Message Board strategy, as this would have the power to send a message to all.

B. Experiments with Physical Hardware

In future we plan to test the cooperation ideas on a small cluster of mobile robots. The simulation approach is enabling us to create swarms with more than one thousand robots; however it would be interesting to compare the simulation results against real life data. For the research we will be using 4 Dr Robot X80-H differential drive style mobile robots. The scenario will mirror the simulations in that the robots will be sent to explore at random. However, for practical purposes, the ‘interesting feature’ or item that they will be searching for will not be as detailed as those used in the simulations. An object or textual sign may be used to represent the item or terrain feature, necessitating the need to use computer vision algorithms for object detection or OCR in the case of a textual sign.

The system will require some additional features such as the use of wheel odometry information so that each robot can accurately localize itself within the environment. This will allow it to work out where it is and move to another robot's position. The robots will start adjacent to each other in a line; the Message Board will receive movement data from each robot and store this in a global system map. The Message Board's map will enable robots to determine where they are in relation to the rest of the swarm. A similar idea was explored in [15], where a central system stored a map that relied on transmitted robot odometry information. The robot would update cells of the map with positive integers to represent a virtual pheromone trail.

For our experiments, each robot will store a local map and use wheel odometry to work out how far they've moved and if they've changed direction. Actual movements such as 'moved 30cm' will need to be translated into pixel movements and coordinates. If the robot moves 30cm, the simulated version on the local map should also move a set number of pixels. This simulated co-ordinate can then be sent to the Message Board when sending or responding to a help request. The location information within a help request will not be more detailed than the simulated versions. The Message Board will filter the help request list based on proximity; the idle robot will only receive the help requests from robots within a certain distance. The idle robot will need to make navigation decisions locally when deciding how to move to the help request co-ordinates.

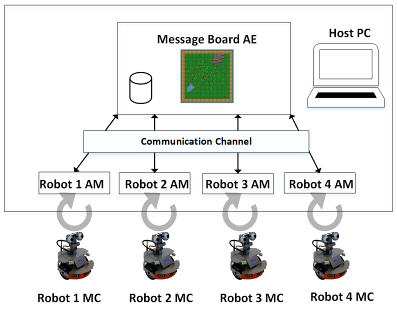

Figure 7. Message Board Autonomic Element and Robot Autonomic Managers running on a PC.

In Figure 7, the system setup is shown, the Message Board Autonomic Element and robot Autonomic Managers all run on the same PC. We may choose to have each running as a separate program and use TCP/IP for interprocess communication; this separation would more accurately mimic a real life scenario. The Message Board AE consists of a Managed Component, which stores all help requests. The MC would also have a virtual Map that could be used to check coordinates and determine proximity when creating a tailored list for an idle robot. Using wheel odometry for localization is flawed and tends to accumulate errors over time. If an idle robot reaches a Help Request location and an odometry error has occurred, it may be necessary to have the robot search the surrounding area for the object.

V. CONCLUSION

This research project aims to apply autonomic computing to the area of swarm collaboration. Instead of replicating the behavior of a natural system, we will create a system were decisions are informed and not reaction based. We believe this type of system would be useful to future space exploration missions where multiple autonomous rovers are sent instead of one all-important rover.

The goal is to simulate multiple direct communication strategies and analyze metrics to determine which strategies perform best in certain situations; a model will then be created from this data. Future work will focus on developing a situational-aware Autonomic Overseer Element that can compare real time data to this model and enable switching between the strategies.

REFERENCES

[1] IBM, "An Architectural Blueprint for Autonomic Computing," IBM, 2003.

[2] R. Sterritt, M. Parashar, H. Tianfield, and R. Unland, "A Concise Introduction to Autonomic Computing," in Advanced Engineering Informatics, 2005, vol. 19, no. 3, pp. 181–187.

[3] W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff, "Autonomous and Autonomic Systems : A Paradigm for Future Space Exploration Missions," IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, 2006, vol. 36, no. 3, pp. 279–291.

[4] W. Truszkowski, H. Lou Hallock, J. Karlin, J. L. Rash, and M. G. Hinchey, Autonomous and Autonomic Systems with Applications to NASA Intelligent Spacecraft Operations and Exploration Systems. London: Springer, 2009.

[5] R. Sterritt, C. Rouff, J. Rash, W. Truszkowski, and M. Hinchey, "Self- * Properties in NASA Missions," in 4th International Workshop on System/Software Architectures (IWSSA'05) in Proc. 2005 International Conference on Software Engineering Research and Practice (SERP'05), 2005, pp. 66–72.

[6] W. Truszkowski, M. Hinchey, J. Rash, and C. Rouff, "NASA's Swarm Missions: The challenge of Building Autonomous Software," IEEE IT Pro, 2004, vol. 6, no. 5, pp. 47–52.

[7] J. C. Barca and Y. A. Sekercioglu, "Swarm robotics reviewed," Robotica, 2012, vol. 31, no. 3, pp. 1–15.

[8] F. Ducatelle, G. A. Di Caro, C. Pinciroli, F. Mondada, and L. Gambardella, "Communication assisted navigation in robotic swarms: Self-organization and cooperation," 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011, pp. 4981–4988.

[9] C. Christodoulopoulos, C. Kyriakopoulos, and A. Kanatas, "A realistic approach to source localization using a wireless robotic network," The Proceedings of First International Conference on Robot Communication and Coordination, 2007, pp. 1–4.

[10] N. Capodieci and G. Cabri, "Collaboration in Swarm Robotics : a Visual Communication Approach," in 2013 International Conference on Collaboration Technologies and Systems (CTS), 2013, pp. 195–202.

[11] A. Anand, M. Nithya, and T. Sudarshan, "Coordination of mobile robots with master-slave architecture for a service application," in 2014 International Conference on Contemporary Computing and Informatics (IC3I), 2014, pp. 539–543.

[12] C. E. Aguero, V. Matellan, J. M. Canas, and V. M. Gomez, "SWITCH ! Dynamic Roles Exchange Among Cooperative Robots," in Proceedings of the 2nd International Workshop on Multi-Agent Robotic Systems - MARS 2006 INSTICC, 2006, pp. 99–105.

[13] M. Veloso and P. Stone, "Video: RoboCup Robot Soccer History 1997 - 2011," in IEEE International Conference on Intelligent Robots and Systems, 2012, pp. 5452–5453.

[14] "Dr Robot X80-H." [Online]. Available: http://www.drrobot.com/products_item.asp?itemNumber=X80-H. [Accessed: 04-January-2016].

[15] I. Susnea, A. Filipescu, V. Minzu, and G. Vasiliu, "Virtual Pheromones and Neural Networks Based Wheeled Mobile Robot Control," in 13th WSEAS International Conference on Systems, 2009, pp. 511–516.