Adaptive Self-Management of Teams of Autonomous Vehicles

Eskindir Asmare, Anandha Gopalan, Morris Sloman, Naranker Dulay, Emil Lupu
Department of Computing, Imperial College London, London SW7 2RH
{e.asmare, a.gopalan, m.sloman, n.dulay, e.c.lupu}@imperial.ac.uk

ABSTRACT

Unmanned Autonomous Vehicles (UAVs) are increasingly deployed for missions that are deemed dangerous or impractical to perform by humans in many military and disaster scenarios. Collaborating UAVs in a team form a Self-Managed Cell (SMC) with at least one commander. UAVs in an SMC may need to operate independently or in subgroups, out of contact with the commander and the rest of the team in order to perform specific tasks, but must still be able to eventually synchronise state information. The SMC must also cope with intermittent and permanent communication failures as well permanent UAV failures. This paper describes a failure management scheme that copes with both communication link and UAV failures, which may result in temporary disjoint sub-networks within the SMC. A communication management protocol is proposed to control UAVs performing disconnected individual operations, while maintaining the SMC's structure by trying to ensure that all members of the mission regardless of destination or task, can communicate by moving UAVs to act as relays or by allowing the UAVs to rendezvous at intermittent intervals.

Categories and Subject Descriptors

C.2.3 [**Network Operations**]: Network management; C.2.4 [**Distributed Systems**]; C.2.8 [**Mobile Computing**]: Mobile environments

Keywords

autonomic management, collaborating autonomous vehicles, delay tolerant networking, communication failure recovery

1. INTRODUCTION

Unmanned Autonomous Vehicles (UAVs) are a type of robot that are often used in civilian disaster relief missions and military scenarios to reconnoiter or provide sensing in areas which are dangerous or impractical for humans. Teams of UAVs may need to cooperate to achieve a particular mission, such as surveillance of a specific area or search for specific targets. A challenge in using UAVs for these missions is enabling adaptive self-management so that they can automatically adapt to changes in context and failures without human intervention. Collaborating UAVs form a Self-Managed Cell (SMC) [11], which is the general architectural principle for realising self management of individual and groups of UAVs. The SMC is an approach to support autonomic computing [7]. A SMC group consists of multiple UAVs and at least one commander, which could be a human or another UAV, to effectively control the group. There could be back-up commanders in case the primary one fails or is lost. The SMC is set up to perform a single mission based on the mission specification received by the commander from its command base. The mission specification defines how UAVs will be assigned to perform specific roles within the SMC, based on their credentials and capability description when they are discovered during the course of the mission.

To ensure that the UAVs comprising the SMC perform their tasks correctly, it is important to cope with different types of failures. Consider a mission scenario that contains the roles: a Commander (C), which has the initial mission specification, assigns roles and manages the SMC; an Aggregator (A), which receives information from surveyors and builds up a map, a Surveyor (S) containing a video camera, a Hazardous-chemical detector (H) that has the required chemical sensor, and a Relay (R) which maintains communication by relaying messages in an ad-hoc network. These roles are initially assigned to the UAVs as a tree, with the commander as the root (as shown in Figure 1). Each UAV will send its state information periodically to its parent node in the tree and in effect the commander will be able to know the relevant membership and activation state of all UAVs in the mission. The commander may backup this state information on other UAVs, one of which may be elected as a commander should the current commander fail. Failures in such missions can occur as a result of communication link failure or individual node failure. The focus of this paper is to detail algorithms and protocols that allow for the correct functioning of a mission even in the presence of failures.

To alleviate the problem arising due to the aforementioned failures, we propose a failure management scheme and a communication management scheme. The failure management scheme is used to cope with the disruptions that occur as a result of intermittent communication link disconnection, permanent communication link failure or failure of one or more UAVs. This scheme uses a management tree similar

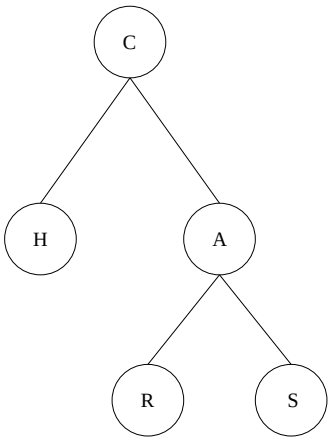

Figure 1: Initial Management Tree consisting of UAVs belonging to a SMC

to the one shown in Figure 1 to define management hierarchies as well as data aggregation hierarchies during execution of the mission. If the periodical state information is not received within a specified length of time (a timeout), it is considered that a failure has occurred. The timeouts are used to differentiate between the types of failures and each failure is handled accordingly.

While the failure management scheme copes with communication link failure and UAV failure, it is sometimes desirable to make sure that the members of the team that form the SMC maintain their communication links. For this purpose, we propose the communication management scheme. Previous works ([1, 3, 12, 9, 10]) ensure this by making sure that the team of robots either restrict their motion to ensure that they do not loose communication or follow each other through line of sight. In our scheme, we follow a two pronged approach. In the first approach, we use a similar technique to the related works mentioned above by restricting the motion of some of the UAVs so as to perform a communication relay function and ensure that communication is not lost with distant UAVs. While this scheme does allow UAVs in a mission to maintain communication links, it is also very restrictive with respect to the motion of the UAV. In the second approach, instead of restricting the motion of the UAVs, we use the concept of a *rendezvous area*, at which the UAVs belonging to the mission can gather at a specified time so that they can exchange the requisite information. In the event that an UAV is unable to reach the *rendezvous area*, it is assumed to have failed and the appropriate failure management mechanism is used.

The rest of this paper is organised as follows. Section 2 details the failure management scheme, while Section 3 details the communication management scheme. Section 4 compares our approach with related work. Section 5 concludes the paper and provides ideas for future work.

2. FAILURE MANAGEMENT

This section presents our failure management scheme which is used to cope with the disruptions that occur as a result of intermittent communication link disconnection, permanent communication link failure or failure of one or more UAVs.

2.1 Management Tree

As a means of decentralising discovery and role management, the UAVs in a mission are arranged in the form of a management tree during the role assignment process. Any of the UAVs in the tree could perform discovery and role assignment. This tree is used for defining management hierarchies as well as for data aggregation during execution of the mission. In this section, we present the algorithm used to form the management tree.

Each UAV runs the tree formation algorithm which starts by broadcasting a discovery message. UAVs receiving the discovery-broadcast perform an authentication protocol if they are not already a member of the SMC and reply with a summary of their capability description if they can be assigned to a role, i.e., UAVs which are already assigned to a role may ignore the broadcast message. Upon authenticating and receiving the capability summary, the broadcaster decides whether to request a full capability description. The authentication protocol is based on the use of public keys but will not be described in this paper. The final decision of assigning the UAV to a role takes place after checking the full capability description against the requirements of the role. If the UAV is assigned to a role, the broadcaster will be the parent for the UAV that replied and the UAV will be listed as a child of the broadcaster (maintained as a list in C and shown in lines 6 and 15 of Algorithm 1). The full steps are shown in Algorithm 1. Algorithm 1 makes use of Algorithm 2 for performing the role assignment.

Algorithm 1 Management Tree Formation Algorithm

MGMT-TREE-FORM(IS_ACTING_CDR)

Input: IS_ACTING_CDR

```
 1: Broadcast DISCOVERY_MSG
 2: Receive MSG
 3: Authenticate sender of MSG
 4: if IS_ACTING_CDR == TRUE then
 5:   if MSG != DISCOVERY_MSG then
 6:     Append ROLE-ASSIGN(MSG) to C
 7:   end if
 8: else
 9:   if MSG == DISCOVERY_MSG then
10:     if PARENT == NULL then
11:       Reply with capability summary
12:       Handle further communications, if any
13:     end if
14:     if MSG == DISCOVERY_REPLY then
15:       Append ROLE-ASSIGN(MSG) to C
16:     else
17:       if MSG == ROLE_ASSIGNMENT then
18:         PARENT = sender of MSG
19:       end if
20:     end if
21:   end if
22: end if
23: return
```

2.2 Failure Detection and Management

We categorise the possible types of failures into intermittent communication link disconnections and UAV failures (per-

Algorithm 2 Role Assignment Algorithm

ROLE-ASSIGN(MSG)

Input: MSG

Output: $CHILD$

1: Check capability summary
2: If the summary is viable, request for full capability description
3: If the description matches the requirement for one of the roles, do role assignment
4: Add the assigned UAV to $CHILD$
5: **return** ***CHILD***

manent communication link failure or failure of one or more UAVs). Timeouts are used to differentiate between the two types of failures. Each UAV periodically sends state information to its parent in the management tree; if the state information is not received within the specified length of time it is considered that a failure has occurred. The timeouts are defined as follows: (a) T_C: the amount of time a parent UAV waits for a state information from a child before it decides that the communication link between itself and the child has failed and (b) T_N: the amount of time a parent UAV waits for a state information from a child before it decides that the child UAV has failed ($T_N > T_C$).

Failure of a communication link and/or a UAV causes partitioning of the team network as well as loss of functionality; we use a systematically defined identity for UAVs to facilitate merging and re-joining of partitioned teams. The identity I of a UAV is defined as: $I = [M \mid H \mid S]$ where: M is the mission ID, H is the hierarchy level and S is a numbering system which puts all the UAVs in the management tree in a total order. This identity lasts throughout the team configuration and identifies the mission and hierarchy level of a UAV in a management tree. The ability to identify this level is useful in handling intermittent link disconnections as discussed in Section 2.3.

2.3 Intermittent Link Disconnection

An intermittent communication link disconnection (failure) may be caused by either a temporary signal blockage by physical objects or movement out of the communication range. Although local functions can keep operating, a temporary partitioning of the logical (overlay) network over which the management tree is formed can cause disruption of state aggregation as well as the flow of management commands. In addition, remote operations will also be affected. The desired response to this type of failure is continuing mission execution with disconnected operations and to resolve inconsistencies when the communication link reappears.

When the team network is partitioned as a result of failure, one or more teams without commanders will be formed. In order to keep the mission execution during the failure, the top UAV on the hierarchy level will become the commander of the team (Note that this UAV was already managing this sub-team during normal functioning). Figure 2 illustrates this process.

A partitioned sub-team can also admit new UAVs. When communication has recovered and the sub-team rejoins the

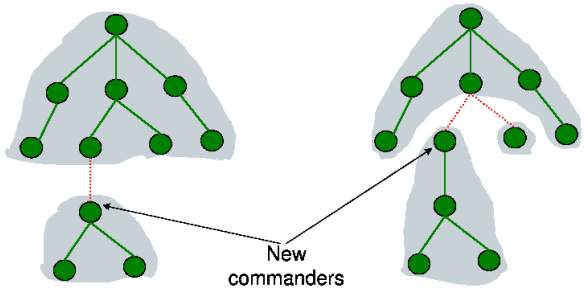

Figure 2: Partitioning due to Link Failure

parent team, the sub-team commander reports its current state to its parent and the domain structure of all UAVs in the mission is updated to indicate new members. To facilitate merging of partitioned teams, we define the hierarchy level of the partitioned team as that of the level of its manager. Merging is performed by placing the lower-level hierarchy teams under the management of higher-level hierarchy teams. Ideally, when there are more UAVs to choose from, more demanding mission subsets (i.e. the ones with more roles to give out) are given to more capable UAVs. Hence, we should keep the more capable UAVs higher up in the management hierarchy.

In this approach, there is no new role assignment in that there is no new UAV assigned to one of the roles or reassignment of existing UAVs to roles different from their original ones. The result being, that the mapping of existing UAVs to roles remains the same whereas the management tree can be different, as it is assumed that the adaptation is temporary. Figure 3 illustrates this approach. The initial configuration is shown in Figure 3 (a). When communication link disconnection occurs, as shown in Figure 3 (b), partitioned sub-teams are created. These sub-teams perform reconfiguration where the partitioned role, H comes under the control of the other sub-team as shown in Figure 3 (c).

2.4 UAV Failure

A UAV failure is caused by either a node failure or a permanent communication link failure. The effect of this type of failure is the partitioning of the team network as well as a loss of roles. The partitioning problem is addressed using the same approach as in the communication link failure. The response to the loss of roles is as follows (in order of priority): (i) use replicated roles, if available, (ii) if there are newly arriving UAVs or previously discovered but unassigned UAVs, perform a role reassignment while keeping the existing team configuration to replace the lost role(s), and (iii) if none of the above is feasible, reconfigure the team by swapping less crucial roles for more crucial roles.

Should the reconfiguration incur role replacement this takes place only in subsets of the team which are lower in hierarchy than the failed UAV. This is based on the assumption that roles assigned to higher hierarchy level UAVs are more crucial to the mission. In the case of role re-assignment and reconfiguration, state information migration takes place.

A permanent communication link failure is also treated as a UAV failure because from the point of view of another UAV it is not possible to infer whether the communication link or the UAV has failed. Figure 3 illustrates how the system adapts to UAV failures. The initial configuration is shown in Figure 3 (a). When a UAV failure occurs, as shown in Figure 3 (d), partitioned sub-teams are created. The response to this problem can be either reconfiguration as shown in Figure 3 (f), where the partitioned sub-teams are moved up in the management hierarchy and now managed by the main commander; or a role replacement where the UAV which was previously assigned to role S is now re-assigned to the supposedly crucial role A as shown in 3 (g).

3. COMMUNICATION MANAGEMENT

In this section, we present our communication management protocol that tries to maintain the communication links between the UAVs involved in the mission. The communication management protocol uses two different approaches. In the first approach, UAVs try and control their movement so as to make sure that they stay within communication range to perform a relay function (Section 3.1). Though this approach allows UAVs involved in a mission to maintain communication links, it would not be feasible in the scenario when UAVs need to reconnoiter. In the second approach (Section 3.2), instead of restricting the motion of the UAVs, we allow UAVs to perform disconnected individual operations while maintaining the SMC structure by trying to ensure that all members of the mission, regardless of destination or task, communicate at intermittent intervals. Before we discuss the algorithms and protocols involved in the communication management scheme, we will list the assumptions: (a) Each UAV knows its current location and also its direction and speed of travel, (b) No clock synchronisation is assumed (though the speed of the clock on the UAVs is assumed to be nearly equal, for example, 20 minutes on one UAV is more or less equal to 20 minutes on another), (c) All UAVs have the same communication range (R) and, (d) A global/local co-ordinate system exists for specifying location and direction of travel.

Consider again the UAV management tree as shown in Figure 1. Each UAV periodically updates its state information to its parent node in the tree and this information is passed onto the commander. For the purpose of our schemes, we augment these periodic messages by the current location and speed of the UAV. This allows the commander to monitor the current position of all the UAVs in the mission. We will now discuss the two approaches in the next two sections.

3.1 Adapt Movement to Maintain Communication

In this section, we detail the approach that controls the movement of the UAVs to ensure that they stay within communication range.

Assume that the position at time T of the 5 UAVs listed in the management tree in Figure 1, are as shown in Figure 4 (a). Starting at time T, UAV S starts to move from its current location to its future location S' with constant speed and direction (Φ). Since the direction and speed of S are available to the rest of the UAVs in the team, it is easy for

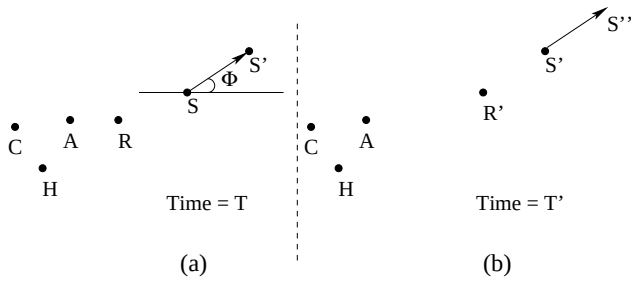

Figure 4: Position of UAVs

them to predict the location of S at a later time (T'). If this position is beyond the communication range of the rest of the UAVs in the mission, the closest UAV to S starts to move in a manner so as to make sure that it still is within communication range. As per the scenario mentioned above, we can see from Figure 4 (a) that UAV R is the closest to UAV S and it is R's job to make sure S is within communication range and it moves accordingly. When S moves to S' at time T', R moves to R' (Figure 4 (b)). The amount that R has to move depends on its location and the location of S. By time T', R moves in a straight line to R', which is the closest point to its current location that is within communication range of S'.

If UAV S keeps moving in the same direction and moves from position S' to S'' during the next time period, then R would also move to keep S within communication range, provided it does not loose communication with the rest of the group. In the event that R along with S move out of communication range with respect to the rest of the UAVs in the group, the UAV closest to R will start following R to keep it within communication range. If S keeps moving away, the rest of the UAVs try and form a "chain" that allows them to keep S within communication range. If it is not possible to cover S, the protocol uses the scheme described in Section 3.2.

3.2 Rendezvous to Restore Communication

In this section, we will detail the approach that allows UAVs to perform disconnected individual operations, while maintaining the SMC structure by trying to ensure that all members of the mission regardless of destination or task, communicate at intermittent intervals.

Consider again the UAV management tree as shown in Figure 1. If the commander UAV notices that the distance between a child node and another member is greater than or equal to the *range threshold* (T_R, which is modelled as a % of the communication range (R)), it initiates the rendezvous algorithm (Algorithm 4). Using the current location, speed and direction of the UAVs in the mission, Algorithm 4 calculates a *rendezvous area* where all the UAVs are expected to rendezvous after a specified time. Once an instance of the rendezvous algorithm is running, future requests are ignored, since the rendezvous area has already been calculated and it is assumed that the newly departing UAV will eventually rendezvous at the same area. After reaching the rendezvous area, the algorithm is restarted only if the need arises again.

Algorithm 4 makes use of another algorithm (Algorithm

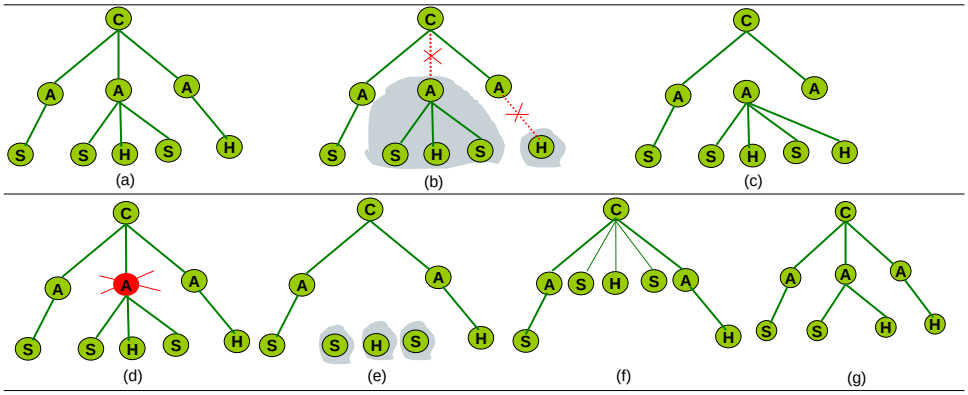

Figure 3: Reconfiguration and Role Re-assignment to Adapt to Failure

3) that actually calculates the rendezvous area. The rendezvous area is calculated as follows. The direction of travel is calculated by averaging the angle of the direction of travel of all the UAVs in the mission with respect to a common axis. Once the direction is calculated, the *rendezvous area* is calculated to be the area surrounding the *rendezvous point* that is achieved by projecting the speed of the slowest UAV starting from the average location onto the direction of travel over the requested time (T). The notations used in the two algorithms are: *REND_MSG*: requests the UAVs to send their current location, speed and direction, n: number of member UAVs that reply to *REND_MSG*, L: list containing the location of the UAVs, $\{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$, V: list containing the speed of the UAVs, $\{v_1, v_2, ..., v_n\}$, A: list containing the direction of the UAVs, $\{\theta_1, \theta_2, ..., \theta_n\}$, v_{min}: minimum speed from amongst $\{v_1, v_2, ..., v_n\}$, θ: average angle of the direction of travel, (X, Y): average location, T: time to rendezvous (this time is relative to current time and indicates the time in the future when the nodes have to rendezvous), (X_{RP}, Y_{RP}): rendezvous point, *REND_AREA*: rendezvous area - a suitable expression for an area around *REND_PT*, and D: distance to rendezvous.

Algorithm 3 Calculate Rendezvous Area

CALC-REND-AREA(L, V, A)
Input: L, V, A
Output: *REND_AREA*

1: $\theta = \frac{\theta_1 + \theta_2 + ... + \theta_n}{n}$

2: $(X, Y) = \begin{cases} X = \frac{x_1 + x_2 + ... + x_n}{n} \\ Y = \frac{y_1 + y_2 + ... + y_n}{n} \end{cases}$

3: $D = T * v_{min}$

4: $(X_{RP}, Y_{RP}) = \begin{cases} 0 \leq \theta \leq \frac{\pi}{2} : \begin{cases} X_{RP} = X + D * sin\theta \\ Y_{RP} = Y + D * cos\theta \end{cases} \\ \frac{\pi}{2} < \theta \leq \pi : \begin{cases} X_{RP} = X - D * sin\theta \\ Y_{RP} = Y + D * cos\theta \end{cases} \\ \pi < \theta \leq \frac{2\pi}{3} : \begin{cases} X_{RP} = X - D * sin\theta \\ Y_{RP} = Y - D * cos\theta \end{cases} \\ \frac{2\pi}{3} < \theta < 2\pi : \begin{cases} X_{RP} = X + D * sin\theta \\ Y_{RP} = Y - D * cos\theta \end{cases} \end{cases}$

5: Calculate *REND_AREA* based on *REND_PT*
6: **return *REND_AREA***

Algorithm 4 Rendezvous Algorithm

REND-ALG()
1: Broadcast *REND_MSG*
2: *REND_AREA* = CALC-REND-AREA(L, V, A)
3: Send the *REND_AREA* and T to the team members
4: **return**

4. RELATED WORK

In [1], the authors suggest a distributed algorithm that allows autonomous mobile robots with limited visibility to converge to a single point. The authors use their previous works in [2, 5] that allows the autonomous mobile robots to agree on a $x - y$ coordinate system, the common origin and the direction of the x-axis. This allows the autonomous mobile robots to exchange their location information and also use this information to converge to a single point. The next position is calculated to be within the smallest region containing the mobile robot and its neighbours.

Similar to [1], the authors in [8, 9] also address the collective behaviour of a group of mobile autonomous agents. The authors discuss two different strategies that allow for mobile autonomous agents to rendezvous at a single specified location (called the *multi-agent rendezvous problem*). Both strategies are "local" strategies, wherein each agent independently calculates its new location based only on its neighbour information. The first set of algorithms consists of strategies that depend on a common synchronised clock between the mobile agents. The second set of algorithms consists of strategies that can be implemented independently, without the need for a synchronised clock.

Though the ideas and protocols provided in the above related works are similar to our idea of a *rendezvous area*, Algorithm 4 is only executed as and when required. Also, we do not restrict the movement of the UAVs since they are free to move in any manner to reach the *rendezvous area*.

In [12, 4], the authors address the communication link problem in a team of mobile autonomous robots by co-ordinating the movement of the mobile robots to keep them within communication range. In [4], the authors use a collection

of robots to follow the "lead" robot as a convoy to ensure that the lead robot does not loose communication link with the base station. The collection of mobile robots act as a relay between the lead robot and the base station and allows the lead robot to carry on with its mission. In [12], the authors develop a Line-Of-Sight communication model and show how the constraints can be optimised for different criteria that depend on the local state of the mobile robot.

Although the ideas suggested in the above works are similar to our approach in Section 3.1, in our approach the robots do not keep following the lead robot, but instead resort to the approach in Section 3.2 to maintain communication links.

In [3, 10], the authors focus on the formation of geometric patterns by a group of mobile autonomous robots. This is helpful in coordinating a group of mobile autonomous robots by allowing them to maintain a pre-agreed upon formation that will ensure that they stay within communication range. In our work, we chose not to take this approach since it is very restrictive with respect to the motion of the robots.

In [13], the authors present a system called Jamp that uses disconnected operations to handle communication link disconnections. They define an abstraction called container, in order to facilitate the implementation of mobile applications. A container is defined as a group of objects and classes that can move into execution environments provided by nodes of the fixed environment or mobile devices. An application in Jamp is implemented as an interaction between containers, since containers can be moved from node to node.

The container concept in the Jamp system and its mobility is similar to our role concept. However, Jamp is not applicable for sudden communication link disconnections since it is not possible to transfer state information to the newly instantiated container in another node. Our approach caters for sudden disconnections by periodically collecting state information by using the management tree.

5. CONCLUSIONS

In this paper, we have proposed algorithms and protocols that allow SMC missions consisting of groups of UAVs to successfully complete in the presence of failures. The approach includes two schemes which augment each other: (a) a failure management scheme that copes with the disruptions that occur as a result of intermittent communication link disconnections, permanent communication link failures or failure to one or more UAVs and (b) a communication management scheme that tries to maintain the communication links between the UAVs involved in the mission.

We are implementing these schemes using the Webots mobile robotics simulator [14] which is a good prototyping environment for modelling, programming and simulating actual mobile robots and then moving the software to the physical robots. Part of the failure management protocol has already been implemented on the Koala robots [6], but since communication range and intermittent disconnections are difficult to control, we are implementing the failure and communication management in the simulator so as to be able to test the working of the protocol with larger groups of robots. The simulator will also allow us to find out the optimal values for *range threshold* (T_R) and the *time to rendezvous* (T).

6. ACKNOWLEDGEMENTS

The work reported in this paper was funded by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.

7. REFERENCES

[1] H. Ando, Y. Oasa, I. Suzuki, and M. Yamashita. Distributed memoryless point convergence algorithm for mobile robots with limited visibility. *IEEE Transactions on Robotics and Automation*, 15(5):818–828, Oct 1999.

[2] H. Ando, I. Suzuki, and M. Yamashita. Formation and agreement problems for synchronous mobile robots with limited visibility. *In Proceedings of the IEEE International Symposium on Intelligent Control*, 1995.

[3] E. Bicho and S. Monteiro. Formation control for multiple mobile robots: A non-linear attractor dynamics approach. *In Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[4] H. Nguyen, N. Pezeshkian, M. Raymond, A. Gupta, J. Spector. Autonomous communication relays for tactical robots. *In Proceedings of the International Conference on Advanced Robotics*, 2003.

[5] I. Suzuki and M. Yamashita. Distributed Anonymous Mobile Robots—Formation and Agreement Problems. *In Proceedings of the 3rd International Colloquium on Structural Information and Communication Complexity*, 1996.

[6] k-team. http://www.k-team.com.

[7] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *IEEE Computer*, 36(1):pp. 41–50, 2003.

[8] J. Lin. Distributed mobility control for fault-tolerant mobile networks. *In Proceedings of Systems Communications*, 2005.

[9] J. Lin, A. S. Morse, and B. D. O. Anderson. The multi-agent rendezvous problem. part 1: The synchronous case. *SIAM J. Control Optim.*, 46(6):2096–2119, 2007.

[10] I. Suzuki and M. Yamashita. Distributed anonymous mobile robots: Formation of geometric patterns. *SIAM J. Comput.*, 28(4), 1999.

[11] J. Sventek, N. Badr, N. Dulay, S. Heeps, E. Lupu, and M. Sloman. Self-managed cells and their federation. In *Workshop Proceedings of the 17th Conference on Advanced Information Systems Engineering.* Springer-Verlag LNCS, 2005.

[12] J. Sweeney, T. Brunette, Y. Yang, and R. Grupen. Coordinated teams of reactive mobile platforms. *In Proceedings of IEEE International Conference on Robotics and Automation*, 2002.

[13] M. Valente, R. Bighonha, M. Bigonha, and A. Loureiro. Disconnected Operation in a Mobile Computation System. *In Proceedings of the Workshop on Software Engineering and Mobility*, 2001.

[14] Webots. http://www.cyberbotics.com. Commercial Mobile Robot Simulation Software.