A Mission Management Framework for Unmanned Autonomous Vehicles

Eskindir Asmare, Anandha Gopalan, Morris Sloman, Naranker Dulay, and 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. UAVs in a team need to operate in sub-groups or independently to perform specific tasks, but still synchronise state information regularly and cope with intermittent communication failures as well as permanent UAV failures. This paper describes a failure management scheme that copes with failures, which may result in disjoint sub-networks within the team. A communication management protocol is proposed to control UAVs performing disconnected individual operations, while maintaining the team's structure by trying to ensure that all members of the mission rendezvous to communicate at intermittent intervals. The evaluation of the proposed approaches shows that the schemes are scalable and perform significantly better than similar centralised approaches.

Keywords: Autonomic management, collaborating autonomous vehicles, mission management, communication failure recovery.

1 Introduction

Unmanned Autonomous Vehicles (UAVs) are mobile robots that are often used in civilian disaster-relief missions and military scenarios to reconnoiter in areas which are dangerous or impractical for humans. A challenge in using UAVs for such 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) [14], a general architectural pattern for realising self management of individual and teams of UAVs. An SMC team consists of multiple UAVs with at least one commander, which could be a human or another UAV. The commander is provided with a mission specification by its command base and assembles the required UAVs to perform the mission. The mission specification [7] defines how specific roles are assigned to certain UAVs based on their credentials and capabilities.

The mission specification also defines a role management hierarchy and the behaviour of these roles in terms of policies specified using the Ponder2 [17] policy specification language. When a mission is instantiated the commander will

```
policy event: /event/newUAV;
condition:[:summary :uav :credential|
  root/discovery uav: uav has: #("motion" "video")
  cap: summary auth: credential];
action: [ :uav |
  root/discovery fullCapReq:uav role:"surveyor"].
```

(a) Initial Role Assignment

```
policy event: /event/UAVFailure;
condition:[:role| role=="surveyor"];
action: [ :role |
  root/role/commander reassign:role
  scheme:"default"].
```

(b) Role Re-Assignment

Fig. 1. Sample Ponder2 Policies

download its role behaviour specifications (the policies) and start the mission. When new UAVs come into the communication range the commander gives a subset of the mission specification to those UAVs which have satisfied the vetting process with respect to capability and credentials. These UAVs may in turn allocate a subset of the mission roles to other UAVs and the whole process finally results in a formation of a management tree which facilitates control and state information collection. Fig. 1 shows examples of role assignment policies. In the first policy the commander authenticates a newly discovered UAV, and assigns it to a surveyor role if it has the required capability with respect to motion and video camera. In the second policy the commander performs reassignment if the failed role type is a surveyor role.

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 following 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, and a Relay (R) which maintains communication by relaying messages in an ad-hoc network. Failures in such missions can occur as a result of intermittent or permanent communication link failures as well as individual node failure. A recent study on UAV failures [4] shows that reliability in field environment is only between 6 and 20 hours.

This paper extends the mission management framework from [6] by evaluating the proposed schemes and elaborating the architecture. This architecture uses a management tree (described in Section 3.1) to define management hierarchies as well as data aggregation hierarchies during execution of the mission. If the periodic state information is not received within a specified timeout period, a failure is considered to have occurred. Various timeouts can differentiate between the types of failures and each is handled accordingly.

In conjunction with failure management, we also actively try to maintain communication between team members using two techniques: i) UAVs adapt their movement to always be within radio range of a neighbour or follow each other (similar to [1,3,19,12,18]) so as to maintain communication by using UAVs as relays to reach distant nodes; ii) The UAVs gather within a defined *rendezvous area* at a specified time so as to exchange the requisite state information (this is due to the fact that it may be impractical to restrict motion in some situations and so we take a *delay tolerant network* approach to cater for UAVs being out of communication range for short periods). In the event that a UAV is unable to reach the rendezvous area, it is assumed to have failed and the appropriate failure management scheme is used. The rendezvous timeout is set to be less than the communication failure timeout so that intermittent disconnection caused by execution of the rendezvous algorithm does not trigger the failure management protocol. It is also possible to change the communication failure timeout dynamically through a policy.

The rest of this paper is organised as follows. Section 2 details the protocol to ensure secure communication within the team. Section 3 details the failure management scheme, while Section 4 details the communication management scheme. Section 5 details the experiments and the ensuing results. Section 6 compares our approach with related work. Section 7 concludes the paper and provides ideas for future work.

2 Security

The UAVs in a team can change over time with new UAVs joining or leaving. These UAVs may also belong to different organisations (e.g. allies). Authenticating a UAV before it joins the team and protecting the ensuing communication is thus necessary to ensure the security of the mission, particularly for military applications. We assume the coalition between different organisations is achieved by using a Central Command Centre (C^3) and use the Certificate Public Key Infrastructure (C-PKI) [9] to ensure authentication, confidentiality and message integrity. The system assumes a single certification authority (C^3), which issues certified public/private keys to all UAVs in the mission and maintains a Certificate Revocation List (CRL). The C-PKI system is also used to exchange a common secret key generated using the Diffie-Hellman protocol [5] between each member of the team and the commander. The secret key effectively establishes a secure channel between the commander and each team member. The steps involved in the authentication between a UAV (A) and the Commander (or any other manager role performing discovery) (C) are shown below:

1. C $\rightarrow$ A: $\{C_{id}\}$. Broadcast Discovery Message.
2. A $\rightarrow$ C: {Join Request, A_{id}, $Nonce_A$}. A sends a request to join the SMC.
3. C $\rightarrow$ A: Sign$\Big\{\{K_c\}_{K_{C^3}^{-1}}, C_{id}, Nonce_A + 1\Big\}_{K_c^{-1}}$. C authenticates itself to A by sending its Public-Key Certificate (PKC) and a function applied to $Nonce_A$, all signed with its private key.
4. A $\rightarrow$ C: Sign$\Big\{\{K_a\}_{K_{C^3}^{-1}}, A_{id}, Nonce_A + 2\Big\}_{K_a^{-1}}$. A sends its PKC to C as well as a function applied to the received Nonce, all signed by its private key. If the certificates are verified by both A and C (using C^3's certificate), mutual authentication is achieved.
5. C $\rightarrow$A: Sign$\{\{g^x \ mod \ p\}_{K_a}, g, p, Nonce_C\}_{K_c^{-1}}$. C sends the Diffie-Hellman parameters and keyshare encrypted with A's public key.
6. A $\rightarrow$ C: Sign$\{\{g^y \ mod \ p\}_{K_c}, Nonce_C + 1\}_{K_a^{-1}}$. A sends its Diffie-Hellman keyshare, encrypted with C's public key.

Both A and C can now calculate a shared secret key (K_{ac}) that is used to establish a secure channel between A and C. The rest of the communication uses the secure channel established above.

3 Failure Management

3.1 Management Tree

The UAVs in a mission are arranged in the form of a management tree during the role assignment process to facilitate decentralisation with any of the UAVs potentially performing discovery and role assignment. This tree is used for defining management hierarchies as well as for data aggregation during execution of the mission. Consider a mission scenario with five UAVs, three hierarchies and five roles in the management tree. Role C is at the top, roles P, Q and R are managed by C and roles S and T are managed by P. Because roles P, Q and R have to be assigned by C before P assigns S and T, C can optimise the assignment by choosing the better suited UAVs for P, Q and R out of the five available UAVs without compromising future assignments because C has a knowledge of future roles to be assigned by P. This is in contrast to a completely distributed task assignment scheme used in architectures such as MURDOCH [8], where decisions are made based only on the current and/or local situation without taking into account how the decision might affect the future and/or global situation. In the following sections we present the management tree formation algorithms. Each UAV, upon start-up runs the algorithm in Section 3.3. However, if the UAV is started as a commander, it runs the *Manager algorithm.* In the event that a UAV becomes a manager, it also runs the *Manager algorithm* (Section 3.2).

3.2 Manager UAV's Algorithm

A manager role has a set of roles it is required to assign according to the mission specification. When the role is started, it prepares a waiting list (W) containing a set of roles to be assigned to UAVs : $W = \{\mathrm{R}_1,\mathrm{R}_2, ..., \mathrm{R}_n\}$ and a children list (L) containing a set of assigned roles and their state information.

1. Broadcast ID periodically to discover other UAVs.
2. If a UAV replies with a join request the manager initiates a mutual authentication process which, if successful, will result in a shared secret key between the managing and managed UAVs. If the authentication is not successful return to step 1.
3. Authenticated UAV sends an encrypted capability summary s, check if there is any role in W with a role assignment policy specifying a capability requirement r, where $r \subseteq s$. If there is such a role then send a request for a full capability description to the UAV.
4. Check if the full capability description satisfies the requirements of the role and if so send a role assignment message to the UAV. Remove the assigned role from W and add it to L.

5. If a state update message is received, update L.
6. Check L for freshness of role state information. If the age of the state of a role is higher than a given interval of time then publish an appropriate failure event (the event could be communication link failure or UAV failure event based on the age of the state). Return to step 5.

For each UAV which has responded to the broadcast, steps 2-4 of the above algorithm execute in parallel. Any UAV can be assigned to a commander role to cater for manager UAV failure.

3.3 Managed UAV's Algorithm

1. Wait for broadcast.
2. If a broadcast message is received and if this UAV can be assigned to a role, send a join request to the broadcaster.
3. If authentication is initiated by the broadcaster, then perform mutual authentication. If the authentication is successful, send an encrypted capability summary to the broadcaster else return to step 1.
4. If a full capability request from the broadcaster is received within a given timeout then send the encrypted description else return to step 1.
5. If a role assignment message is received within a given timeout then download the policies specifying the behaviour of the role, start the role and identify the broadcaster as the parent (manager) UAV else return to step 1.
6. Send a state update message to the manager UAV periodically.

Fig. 2 illustrates a trace of the tree formation algorithms. Fig. 2(a) shows the communication links between neighbouring nodes. In Fig. 2(b) the top node broadcasts Discovery messages to its neighbours which form a team with the top node as commander and the middle nodes as children assigned to various roles (Fig. 2(c)). In Fig. 2(d), the middle nodes broadcast to their neighbours but only lower nodes respond as the other middle nodes already have a parent. Fig. 2(e) shows the resulting tree with a single parent for each node.

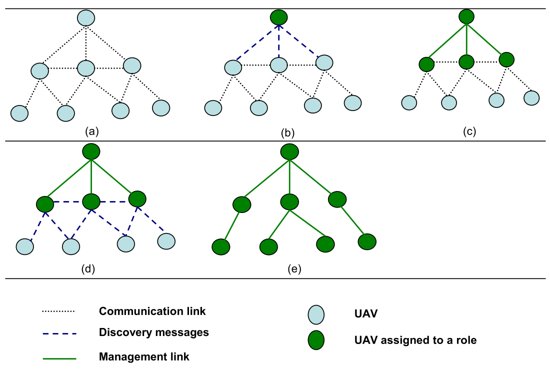


Fig. 2. Management Tree Formation

3.4 Failure Detection and Management

We use different timeouts to distinguish between intermittent communication link failures and permanent communication link or UAV node failures. Each UAV periodically sends state information to its parent in the management tree; if the state information is not received within a specified timeout it is considered that a failure has occurred. The timeouts are: (a) T_C: detects intermittent communication link failure (b) T_N: detects permanent failures ($T_N > T_C$).

Failure of a communication link and/or a UAV causes partitioning of the team 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 = mission ID, H = hierarchy level and S = a numbering system to place all the UAVs in the management tree in a total order. This identity lasts throughout the team configuration.

3.5 Intermittent Link Failure

An intermittent communication link 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 to continue mission execution with disconnected operations and resolve inconsistencies when the communication link reappears.

When the team 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 (which was already managing this sub-team during normal functioning) will become the commander of the team. A partitioned sub-team can also admit new UAVs. When the sub-team rejoins the 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 to be the level of its manager. Merging is performed by placing 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 (ones with more roles) are given to more capable UAVs. Hence, we should keep more capable UAVs higher up in the hierarchy.

In this approach, there is no new role assignment 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. The initial configuration is shown in Fig. 3(a). When communication link disconnection occurs, as shown in Fig. 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 Fig. 3(c).

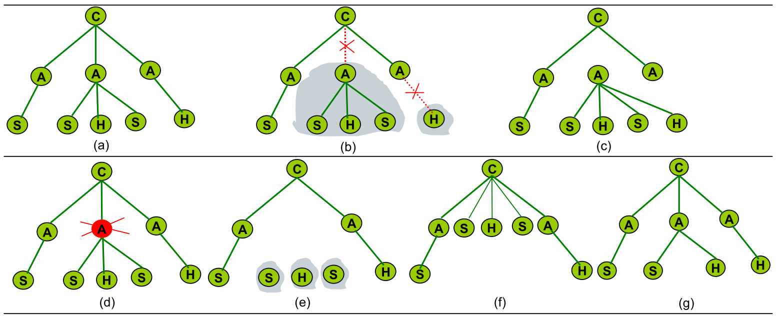


Fig. 3. Reconfiguration and Role Reassignment to Adapt to Failure

3.6 Permanent Failures

A permanent failure is caused by either a node or communication link hardware failure (other UAVs cannot distinguish between these). The result is the partitioning of the team as well as a loss of roles. The partitioning problem is addressed using the approach in Section 3.5. The response to the loss of roles is as follows (in order of priority): (i) use replicated roles, if available, (ii) if there are unassigned or newly discovered 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 due to the fact that roles assigned to higher level UAVs are more crucial to the mission. In the case of role reassignment and reconfiguration, state information migration takes place.

Fig. 3 illustrates adaptation to permanent failures. The initial configuration is shown in Fig. 3(a). When a permanent failure occurs, as shown in Fig. 3(d), partitioned sub-teams are created (Fig. 3(e)). The response can be either reconfiguration as shown in Fig. 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 reassigned to the supposedly crucial role A as shown in 3(g). All reconfigurations, reassignments and other responses are specified in terms of policies.

4 Communication Management

In this section, we present our communication management protocol that tries to maintain the communication links between the UAVs in the mission in order to prevent communication link failure. We assume: (a) Each UAV knows its current location and its direction and speed of travel, (b) No clock synchronisation, but relative time is assumed to be consistent i.e. 20 minutes on one UAV is approximately equal to 20 minutes on another, (c) All UAVs have the same communication range (C_R) and, (d) A global/local co-ordinate system exists for specifying location and direction of travel. For the purpose of our schemes, we augment the periodic state update messages (sent between the UAVs, as specified by the management tree) by the current location and speed of the UAV.

4.1 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.

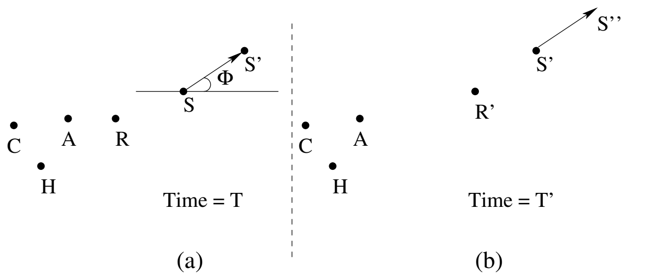


Fig. 4. Position of UAVs in the mission

Assume that the position at time T of the 5 UAVs in the mission are as shown in Fig. 4(a). At time T, UAV S starts moving 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 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 of S. As per the scenario mentioned above, we can see from Fig. 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' (Fig. 4(b)). The amount that R has to move depends on its location and the location and speed of S. If S moves from position S' to S'' during the next time period, then R would also move to keep S within communication range. 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 4.2.

4.2 Rendezvous to Restore Communication

Though the approach detailed in Section 4.1 allows UAVs involved in a mission to maintain communication links, it would not be feasible in the scenario when

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

If the commander UAV notices that the distance between a child node and another member is greater than the *range threshold* (T_R, modelled as a % of the communication range (C_R)), it initiates the rendezvous algorithm. Using the current location, speed and direction of the UAVs in the mission, the *rendezvous area* is calculated and communicated to the team members. This is 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. After reaching the rendezvous area, the algorithm is restarted only if the need arises again.

The rendezvous area is calculated as follows. The average direction of travel (θ) 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 (using a suitable expression) surrounding the *rendezvous point* that is achieved by projecting the speed of the slowest UAV starting from the average location (X, Y) onto the average direction of travel over the requested time (T, which is relative to current time and indicates the future time when the nodes should rendezvous). The rendezvous point is calculated as follows (D = distance to rendezvous):

$$X_{RP}, Y_{RP} = \begin{cases} X_{RP} = X + D * cos\theta \\ Y_{RP} = Y + D * sin\theta \end{cases} \tag{1}$$

5 Experiments and Results

The experimental setup consisted of machines on a Local Area Network. We simulated different subnets by using IP filter policies. Each manager role was assigned to a separate machine and a different subnet, while other roles were running in parallel (with a maximum of 20 roles per machine).

5.1 Mission Setup Time

In this experiment we fixed the depth of the management tree to 5 levels and compared its performance, with respect to mission setup time, with a centralised approach by varying the number of roles in the mission. Fig. 5(a) shows the result for 25 experiments plotted with a 95% confidence interval. The result illustrates that as the number of roles increases the hierarchical management approach outperforms the centralised one.

5.2 Effect of Depth of the Management Tree and Number of Roles

In this experiment we varied the number of roles between 20 and 200 and the depth of the management tree between 1 and 10. We then measured the mission

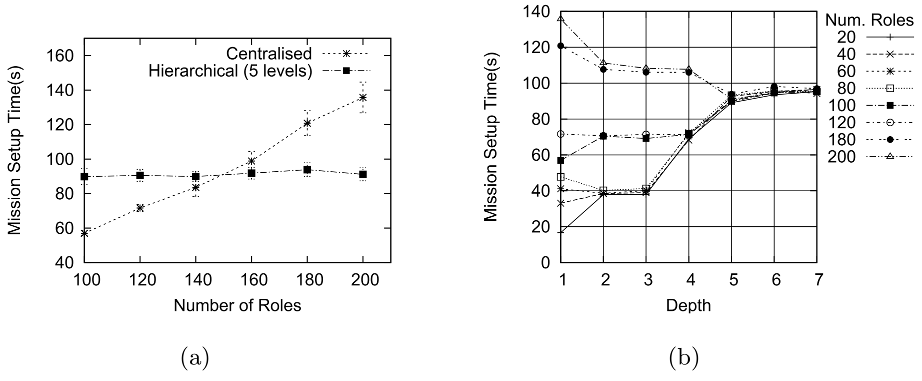


Fig. 5. Mission setup time

setup time. Fig. 5(b) shows the result for upto depth 7 (after depth 7, there is very little change to the mission setup time with respect to the depth) for 25 experiments. For higher number of roles, the mission setup time decreases as we increase the depth of the tree as a result of load balancing. However this trend stops and the setup times start to increase slowly as the tree becomes deeper due to the delay in role assignment created by an increase in the number of hops. This behaviour suggests the existence of a ratio of number of roles to depth, for a given management tree, which guarantees a minimal mission setup time. For smaller number of roles the mission setup time is minimal at depths 1 and 2 after which the setup time increases with depth due to an overhead introduced by the added number of hops without any gain in load balancing as the number of roles managed is already sufficiently small. It is also interesting to note that the depth at which the minimal setup time occurs increases as we increase the number of roles.

5.3 Mean Time to Reassign Roles After Failure

In this experiment we study the response time of our mission management system when a cluster of failures occur, as in typical disaster response or military scenarios it is likely that a group of UAVs could be affected by an event causing

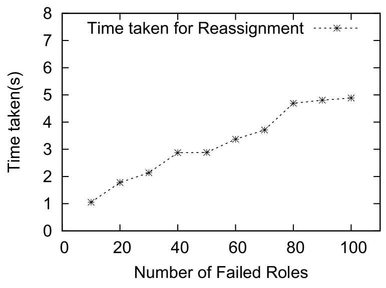


Fig. 6. Measurement of Time Taken to Reassign Roles in a Cluster Failure Scenario

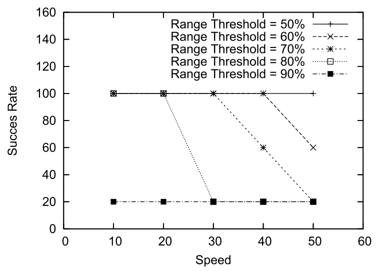
them all to fail. We used a mission specification which has 100 Surveyor roles and 100 Aggregator roles and a reassignment policy which dictates that whenever a Surveyor role fails an Aggregator role should be withdrawn from a working UAV and replaced by a Surveyor role. The results are shown in Fig. 6. We note that the reassignment time scales linearly with the number of failed nodes.

5.4 Evaluation of Communication Management

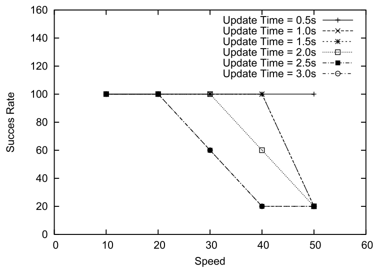
The communication management scheme was implemented using the Webots mobile robotics simulator [15], which is a prototyping environment for modelling, programming and simulating mobile robots. In the first experiment (Fig. 7(a)), the effect of range threshold was evaluated with respect to the speed of the UAVs, while the second experiment (Fig. 7(b)) evaluated the update time versus the speed of the UAVs. The success rate is defined as the number of UAVs that successfully manage to follow the lead UAV to its destination (including the lead UAV itself). A total of five UAVs were used for this experiment and they were arranged in a management tree with one commander, an aggregator, a surveyor, a hazard detector and a relay. For the purpose of the experiments, the relay was acting as the "lead" UAV. For the value of speed, a magnitude of 1 denotes a speed of 4.5 mm/s. For the first experiment, the update time is set to $2s$, while for the second experiment, the range threshold is set to 75%.

From Fig. 7(a), we see that the range threshold (T_R) has a significant impact on the performance. As the value of T_R increases, fewer and fewer UAVs are able to follow the leader. This is especially true in the case when the speed is greater than 30, since only the leader is able to reach its destination. Setting the value of T_R much lower (50%) enables everyone to reach the destination, but this may be detrimental since this results in the leader being followed very closely and may result in a cluster failure (due to a hazardous terrain). Instead of setting the value of T_R apriori, it is ideal to set the value dynamically based on the current speed. For lower speeds, a high value of T_R would suffice and vice versa.

From Fig. 7(b), we can see that the change in update time adversely affects the UAVs when they are travelling at a high speed. This is to be expected since



(a) Range-Threshold



(b) Update-Time

Fig. 7. Communication Management

the "follower" UAV uses the location updates of its leader to map its path. Having a small update rate ($0.5s$) results in all UAVs following the leader to the destination. However, this has an adverse effect on the battery life of the UAVs due to the excess communication. In case we wish to improve the performance we can always decrease the value of T_R, rather than increasing the update time.

6 Related Work

A distributed algorithm that allows autonomous mobile robots with limited visibility to converge to a single point is suggested in [1]. This uses their previous work in [2,11] that allows the mobile robots to agree on a $x-y$ coordinate system, the common origin and the direction of the x-axis. This allows the mobile robots to exchange their location information and use this information to converge to a single point. Similarly [13,12] also addresses the collective behaviour of a group of mobile autonomous agents and discusses two different strategies that allow for these to rendezvous at a specified location. Both strategies are "local" strategies, wherein each agent independently calculates its new location based only on its neighbour information. The ideas and protocols provided above are similar to our idea of a *rendezvous area* but our rendezvous algorithm 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*.

Co-ordinating the movement of the mobile robots to keep them within communication range is discussed in [19,16]. Although the ideas are similar to our approach in Section 4.1, our robots do not keep following the lead robot, but instead resort to the approach in Section 4.2 to maintain communication links.

In [10] the authors present an approach for exploration, mapping and tracking targets using large scale heterogeneous robots. They classify the robots based on their capabilities as highly-capable, slightly less-capable and simple. The less capable robots (leader robots) are responsible for leading the simple robots as the simple robots do not have the navigation capability. The failure detection approach is slightly similar to our approach in that the leader robot detecting the failure of the follower is comparable to a parent role detecting the failure of its child. However, their approach to recovery is coded in the behaviour of the robots while we use policies for managing failure. Also, whereas we use disconnected operations in communication failure and reassignment in complete UAV failure; in their approach the leader robots return home when communication failure occurs and there is no defined action for a complete leader robot failure.

Jamp [20] uses disconnected operations to handle communication link disconnections and defines an abstraction called container, in order to facilitate the implementation of mobile applications. 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 communication link failures since it is not possible to transfer state information to the newly instantiated container in another node. Our approach caters for link disconnections by periodically collecting state information by using the management tree.

7 Conclusion

In this paper we have presented a failure management scheme for teams of Unmanned Autonomous Vehicles performing a mission using a hierarchical mission management approach. We evaluated the performance of both the hierarchical mission management system and the failure management scheme. The results show that the mission management system is scalable and also has the advantage of having a shorter mission setup time, especially for large scale missions, as compared to a centralised management approach. The failure management scheme scales linearly with the failed number of roles which makes it suitable for large scale failures in difficult or dangerous mission areas.

We have also presented a communication maintenance scheme that tries to maintain the communication links between the UAVs involved in the mission. The proposed scheme was evaluated in the Webots simulator and the ensuing results show that the scheme is robust and flexible.

Future work will focus on studying the response of the failure management scheme by using different failure models. We also intend to study reconfigurations of the mission management tree triggered by withdrawal of a role during a reassignment. Also, the approach to maintain communication using the rendezvous algorithm will be implemented on the Webots simulator and evaluated.

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.

References

1. Ando, H., Oasa, Y., Suzuki, I., Yamashita, M.: Distributed memoryless point convergence algorithm for mobile robots with limited visibility. IEEE Transactions on Robotics and Automation 15(5), 818–828 (1999)
2. Ando, H., Suzuki, I., Yamashita, M.: Formation and agreement problems for synchronous mobile robots with limited visibility. In: Proceedings of the IEEE International Symposium on Intelligent Control (1995)
3. Bicho, E., Monteiro, S.: 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. Carlson, J., Murphy, R.R.: How UGVs physically fail in the field. IEEE Transactions on Robotics 21(3), 423–437 (2005)
5. Diffie, W., Hellman, M.: New directions in cryptography. IEEE Transactions on Information Theory 22(6), 644–654 (1976)
6. Asmare, E., Gopalan, A., Sloman, M., Dulay, N., Lupu, E.C.: Adaptive Self-Management of Teams of Autonomous Vehicles. In: Proceedings of the 6th International Workshop on Middleware for Pervasive and Ad-Hoc Computing (2008)

7. Asmare, E., Gopalan, A., Sloman, M., Dulay, N., Lupu, E.C.: A Policy-Based Management Architecture for Mobile Collaborative Teams. In: Proceedings of the 7th Annual IEEE International Conference on Pervasive Computing and Communications (2009)
8. Gerkey, B.P., Mataric, M.J.: Principled communication for dynamic multi-robot task allocation. In: ISER 2000: Experimental Robotics VII, London, UK, pp. 353–362. Springer, London (2001)
9. Housley, R., Ford, W., Polk, W., Solo, D.: Internet X. 509 Public Key Infrastructure Certificate and CRL Profile. Technical report, RFC 2459 (January 1999)
10. Howard, A., et al.: The sdr experience: Experiments with a large-scale heterogenous mobile robot team. In: 9th International Symposium on Experimental Robotics (2004)
11. Suzuki, I., Yamashita, M.: Distributed Anonymous Mobile Robots—Formation and Agreement Problems. In: Proceedings of the 3rd International Colloquium on Structural Information and Communication Complexity (1996)
12. Lin, J., Morse, A.S., Anderson, B.D.O.: The multi-agent rendezvous problem. part 1: The synchronous case. SIAM J. Control Optim. 46(6), 2096–2119 (2007)
13. Lin, J.: Distributed mobility control for fault-tolerant mobile networks. In: Proceedings of Systems Communications (2005)
14. Lupu, E., et al.: AMUSE: Autonomic management of ubiquitous e-health systems. Concurrency and Computation: Practice & Experience 20(3), 277–295 (2008)
15. Michel, O.: Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems 1(1), 39–42 (2004)
16. Nguyen, H., et al.: Autonomous communication relays for tactical robots. In: Proceedings of the International Conference on Advanced Robotics (2003)
17. Ponder2, `http://ponder2.net`
18. Suzuki, I., Yamashita, M.: Distributed anonymous mobile robots: Formation of geometric patterns. SIAM J. Comput. 28(4) (1999)
19. Sweeney, J., et al.: Coordinated teams of reactive mobile platforms. In: Proceedings of IEEE International Conference on Robotics and Automation (2002)
20. Valente, M., Bighonha, R., Bigonha, M., Loureiro, A.: Disconnected Operation in a Mobile Computation System. In: Proceedings of the Workshop on Software Engineering and Mobility (2001)