UAV Swarm Coordination using Cooperative Control for establishing a Wireless Communications Backbone

Achudhan Sivakumar
School of Computing
National University of Singapore
13 Computing Drive, Singapore
achudhan@nus.edu.sg

Colin Keng-Yan Tan
School of Computing
National University of Singapore
13 Computing Drive, Singapore
colintan@nus.edu.sg

ABSTRACT

In this paper, we present a mechanism to fly a swarm of UAVs with the aim of establishing a wireless backbone over a pre-specified area. The backbone is aimed at connecting intermittent wireless-signal-emitting mobile ground stations (GSs), comprising rescue teams and survivors. To this end, we present a decentralized behavior-based cooperative control architecture to search for unknown GSs and relay packets from one GS to another. A delay tolerant network protocol implementation is assumed on the agents but maintained transparent to the GSs. The conditions for agent state transition are adapted to maximize a measured performance score. A novel belief exchange mechanism for cooperation is designed to utilize low bandwidth through state estimation by a Dynamic Cell Structure. Extensive simulations are performed to prove the effectiveness of the proposed solution via measured performance metrics like average latency and visit frequency.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Coherence and coordination, Multiagent systems*

General Terms

Design, Experimentation, Performance

Keywords

cooperative control, disaster relief, DTN, swarm coordination, teamwork

1. INTRODUCTION

The response phase in a disaster management situation plays a key role in mitigating possible adverse effects including loss of lives. Part of the response phase involves the dispatch of rescue teams (on ground) into the disaster area to survey the damage and find survivors. These rescue teams often need to send data back to the base station or to other rescue teams in the area. Data could include information like images, videos, or even calls for additional support. Moreover, communication between the rescue teams and with the base station can greatly enhance coordination between the various teams. Unfortunately, in a disaster situation, normal communication infrastructure tends to be damaged or destroyed. Traditionally, push-to-talk services [6] have been used for voice communication between base stations and rescue teams. However, such services are not designed to handle data communications involving images, videos, sensor readings, etc., that require higher bandwidths. Attempts have been made to use satellite communications for exchange of information between first responders [11]. However, satellite communications through services like Iridium [9] provide very low bandwidths in the range of 10kbps. This scarce bandwidth would have to be shared by multiple rescue teams in the same area, thus making the available bandwidth for each team, extremely small.

A feasible alternative is to deploy a set of unmanned aerial vehicles (UAVs), each mounted with a wireless communication device like a WiFi antenna, so as to build a wireless backbone over which various entities on the ground such as rescue teams, relief agencies, survivors, first responders, etc. can communicate. In fact, it has been proven through real experiments that air-to-ground communication through commercial off-the-shelf (COTS) 802.11 equipment is viable [7]. In this solution, a system of aircrafts would provide a mobile ad hoc network (MANET) connecting ground devices like laptops, PDAs, cell phones, and devices capable of wireless communication. One plausible approach is to maintain a fully connected network of UAVs at all times. For example, in [8], a chain of UAVs is maintained at all times so that a given UAV may communicate with any other UAV using multi-hop ad hoc routing. Correspondingly, numerous ad hoc routing protocols have been proposed for rapidly changing network configurations [10]. However, often times there aren't enough UAVs to establish a continuous link between two points on the ground and this is a huge problem for solutions that require a fully connected UAV mesh. The notion of a continuous link between end-points is meaningful when the relays are stationary. Mobile relays on the other hand, can act as ferries to deliver packets, thus eliminating the need for a continuous link. In order to support such mobile relays, a new class of routing protocols have emerged for what are known as Delay Tolerant Networks (DTNs). The concept of delay tolerance is directly applicable when using UAVs as communication relays. Research on DTNs has been extensive over the last few years. However, lit-

Cite as: UAV Swarm Coordination using Cooperative Control for establishing a Wireless Communications Backbone, A. Sivakumar and C. K. Y. Tan, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. . 1157-1164

tle work has been done on cooperatively controlling UAVs to physically establish such DTNs. A few fixed trajectory solutions have been proposed that utilize DTNs for establishing communication between two mutually unreachable ground stations [16, 5]. They use one of two configurations: chain-relay or conveyor belt. However, neither is scalable for scenarios with a number of ground stations scattered in a given area. In fact, these fixed trajectory solutions would cease to work with the introduction of mobile ground stations. Moreover, if communication is to be established for survivors too, searching the area would be a necessity that cannot be achieved with fixed trajectory solutions.

We propose the use of autonomous agents on-board each UAV to cooperatively control them and form a multi-agent system that achieves the goal of establishing communication between multiple ground stations. To this end, we give a detailed description of the system that determines the behavior of each agent and the mechanism used to coordinate their actions. The approach we take is a very practical one with all simulations performed on a realistic flight simulator (X-Plane) and wireless communication simulator (Qualnet). In other words, the solution proposed works under the constraints of UAV aerodynamics and wireless range limitations. Since there is one agent associated with each UAV, we shall be using the terms agent and UAV interchangeably.

2. PROBLEM DESCRIPTION

In this paper, we assume an $M \times N$ grid representing the disaster struck area. A set of K agents (UAVs) are dispatched to start operations from random positions in the grid. To account for the worst case scenario, agents are assumed not to have *a priori* information about positions of rescue teams and survivors. The agents however have knowledge of the base station's position, which is along one of the edges of the grid, bordering the disaster struck area. All ground stations are allowed to move, as long as they remain within the boundaries of the grid. Ground stations are assumed to have a maximum speed of $10ms^{-1}$ and are not expected to transmit signals all the time. In other words, a given ground station could be an intermittent signal emitter. The task of the agents then is to constantly keep searching for ground stations, and establish communications between those that have already been found. The challenge is to maximize both search frequency (of the entire area) as well as bandwidth available to each ground station while minimizing the latency for packet delivery. The bandwidth available to a ground station can be considered to be directly proportional to the amount of time it has an agent within communication range, also known as service duration. Since every GS has only one transmitting radio and since the data rate is capped by the constant wireless link capacity, the only variable that affects the amount of data transmitted per unit time is service duration. As a result, the aim is to maximize quality, Q, of the search and relay operation,

$$Q = \frac{f_{avg} s_{avg}}{l_{avg}} \quad (1)$$

where f_{avg} = average visit frequency for all grid cells measured over the last T_W time units
$= \frac{1}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} f_{i,j}$

s_{avg} = average service time over the last T_W time units for ground stations
$= \frac{1}{G} \sum_{i=1}^{G} s_i$

l_{avg} = average latency for all packets delivered in the last T_W time units
$= \frac{1}{P} \sum_{i=1}^{P} l_i$

System-wise, every UAV is assumed to be equipped with a GPS receiver for position information. Each UAV is also mounted with an omni-directional WiFi antenna with a transmitting power of 20dBm, which gives a theoretical communication range of 350m. We assume a practical communication range of 150m air-to-ground and 200m air-to-air. This allows us to divide the test area of 2km×2km into a 20×20 grid so that even if the UAV were to fly through one of the corners of a grid cell, the whole cell would be covered, owing to a cell diagonal length of 140m.

3. COOPERATIVE CONTROL ARCHITECTURE

The control architecture we propose is a distributed one, wherein each agent makes control decisions independently, using information from own sensors and from communicating with neighboring agents. Each agent's belief of the world is represented using two data structures:

1. A grid of integers called the Visit Map (VM): The value assigned to each grid cell is the time elapsed since any agent last visited that grid cell. This is different from the belief map that many other papers on multi-agent target search use. In many works, the belief map is a probability distribution giving the probability of finding a target in a given grid cell[14]. We however use elapsed time in order to enable the hybrid state where an agent performs the search operation as well as relaying of packets between ground stations. The behavior of an agent in this hybrid state is detailed in section 3.3. The value for each cell is incremented by 1 at every time step. When the agent flies over a particular grid cell, the corresponding value for that cell in the VM is reset to 0.

2. A set of currently known positions (in grid coordinates) for ground stations, called the Position List (PL): This PL is updated when the agent detects a wireless signal from a ground source when flying over a grid cell. It is also updated when information is received from neighboring agents.

Belief information is exchanged between agents when they come within communication range of each other. Both, the VM as well as PL are exchanged between neighboring agents. The details of what entails an information exchange is provided in section 3.6.

Every agent, at its core is a behavior-based control system. At the higher level though, every agent is capable of operating in one of the following 4 states: search (SR), relay (RL), search and relay hybrid (HB), and proxy (PR). We now take a look at the behavior of an agent in each of the 4 states.

3.1 Search State

In the search state, an agent flies around the entire grid looking for wireless signals from a ground source. We do not use predetermined search patterns like those mentioned in [12] so that UAVs can be added or removed from the multi-agent system at any time. This is essential because in a realistic setting, UAVs may run out of power or fuel, or may suffer failures in mid-air that require their withdrawal and replacement. In other words, an agent's control decision cannot depend on the knowledge of number of UAVs on the field. Moreover, since ground stations can emit signals intermittently, the search operation can never come to an end. In other words, the agents need to revisit each grid cell repeatedly through time. Keeping these requirements in mind, we aim to design a cooperative search mechanism where behaviors of each agent combine to provide an emergent behavior wherein the multiagent system spreads out to search the entire area.

At every instance when an agent moves from one grid cell to another, it recomputes its action. When an agent wants to compute its action, it assigns a score to every cell in the grid using the formula in Equation 2, and picks the cell with the highest score as its next destination.

$$w_{k_t} t_{k_{ij}} + w_{k_h} H(h_{k_{\text{pref}}} - h_{k \to ij}) + w_{k_d} G(d_{k_{ij}} - d_{k_{opt}}) \quad (2)$$

The above formula gives $\text{score}_{k_{ij}}$, which is the score assigned by agent k to grid cell (i, j). It is computed as a summation of three components. The first is $w_{k_t} t_{k_{ij}}$ where w_{k_t} is a positive weight and $t_{k_{ij}}$ is the elapsed time value for grid cell (i, j) held in VM_k. This term represents the desire of each agent to visit the grid cell that has not been visited in the longest while, i.e. the grid cell with the highest value on VM_k. This is essential because the validity of information about a cell decays with time. As a result, the cell with the highest elapsed time, is the one about which there is least certain information. The second term in Equation 2 is $w_{k_h} H(h_{k_{\text{pref}}} - h_{k \to ij})$ where w_{k_h} is a positive weight and $h_{k \to ij}$ is the heading from agent k to the center of grid cell (i, j). $h_{k_{\text{pref}}}$ refers to the preferred heading of agent k and is given by

$$h_{k_{\text{pref}}} = \begin{cases} h_{k_{curr}} & \text{if } N(k) = \emptyset \\ \frac{1}{|N(k)|} \sum_{i \in N(k)} h_{i \to k} & \text{otherwise} \end{cases} \quad (3)$$

where $h_{i \to k}$ is heading from agent i to agent k
$N(k) = \{i |$ agent i is in range of agent $k\}$

Equation 3 means that the preferred heading of an agent is its current heading, unless there are neighboring agents within communication range. Essentially, every agent desires to move forward where forward is defined as any direction falling within $\frac{\pi}{4}$ radians from the current heading as can be visualized in Figure 1. In the presence of neighbors, the preferred heading of an agent is the average of the set of headings away from every neighboring agent. This mechanism mainly achieves the spread of agents in opposite directions. As a side effect, it also achieves collision avoidance. However, collision avoidance is not considered explicitly in this paper and therefore cannot be guaranteed. There are algorithms that have been proposed to guarantee collision avoidance using multiple altitudes that can be applied here if required. The difference between $h_{k_{\text{pref}}}$ and $h_{k \to ij}$ is fed as input to function, $H(x)$, which is a combination of two

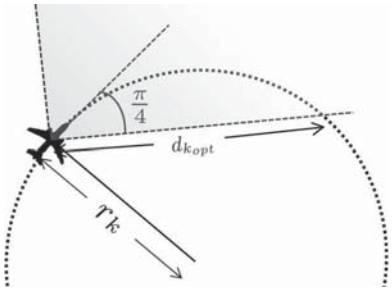

Figure 1: Optimal distance, $d_{k_{opt}}$

Gaussian functions given by

$$H(x) = \begin{cases} e^{-\frac{x^2}{2}} & \text{if } |x| \leq \frac{\pi}{4} \\ 5e^{-\frac{x^2}{c}} & \text{if } |x| > \frac{\pi}{4}. \end{cases} \quad (4)$$

$$\text{with} \qquad c = \frac{2\pi^2}{32 \ln 5 + \pi^2}$$

As a result, the second term in Equation 2 has the highest value when the difference between $h_{k_{\text{pref}}}$ and $h_{k \to ij}$ is 0 and gradually decreases until the magnitude of this difference hits $\frac{\pi}{4}$. When the magnitude of the difference increases beyond $\frac{\pi}{4}$, the value of the second term in Equation 2 drops steeply towards 0. All in all, the second term represents the desire of an agent to continue flying along the preferred heading (preferably within $\frac{\pi}{4}$ radians of $h_{k_{\text{pref}}}$) which would be plain forward in the absence of neighboring agents and away from all neighboring agents, if there were any.

The third term in Equation 2 represents the fact that each agent in general concerns itself with the cells closest to it and gives them more importance as compared to cells that are farther. It is given by $w_{k_d} G(d_{k_{ij}} - d_{k_{opt}})$ where w_{k_d} is a positive weight and $d_{k_{ij}}$ is the distance of agent k from the center of grid cell (i, j). $d_{k_{opt}}$ is given by

$$d_{k_{opt}} = \sqrt{2} r_k \quad (5)$$

where $r_k =$ minimum turn radius of agent k

In order to understand the reason behind Equation 5, let us imagine a situation where agent k is at the center of grid cell $(5, 5)$ and heading towards $(5, 6)$. The score for grid cell $(5, 7)$ would be high even though it might be impossible to reach $(5, 7)$ with a simple bank maneuver owing to the minimum turn radius of a UAV. As a result the optimal distance to look ahead for an agent should be the distance of the intersection of the UAV's trajectory with maximum bank angle and the line emanating from the agent at an angle of $\frac{\pi}{4}$ from the current heading as shown in Figure 1. The angle of $\frac{\pi}{4}$ is chosen so as to fall in line with the definition of forward. The difference between $d_{k_{ij}}$ and $d_{k_{opt}}$ is fed as input to the function $G(x)$, which is a Gaussian function given by

$$G(x) = e^{-\frac{(x)^2}{2\sigma^2}} \quad (6)$$

$$\text{with} \qquad \sigma = \frac{d_{max}}{\sqrt{2 \ln (100)}}$$

where d_{max} is the maximum possible distance between two points on the grid (the diagonal if it is a rectangle). As a

result, the third term peaks when $d_{k_{ij}} = d_{k_{opt}}$ and drops as the difference between the two increases.

The three weights w_{k_t}, w_{k_h}, and w_{k_d} are chosen based on the following rules:

1. Considering cells (i_1, j_1) and (i_2, j_2) at a fixed distance from the agent, if $h_{k_{\text{pref}}} - h_{k \to i_1 j_1} = 0$ and if $h_{k_{\text{pref}}} - h_{k \to i_2 j_2} = \frac{\pi}{4}$, then a difference in elapsed time of $t_{k_{i_2 j_2}} - t_{k_{i_1 j_1}} = C_t$ should make the scores for both equal.

2. Considering cells (i_1, j_1) and (i_2, j_2) with the same elapsed time value on VM_k, if $h_{k_{\text{pref}}} - h_{k \to i_1 j_1} = 0$ and $h_{k_{\text{pref}}} - h_{k \to i_2 j_2} = \frac{\pi}{4}$ and $d_{k_{i_2 j_2}} = d_{k_{opt}}$, then a difference in distance of $d_{k_{i_1 j_1}} - d_{k_{i_2 j_2}} = C_d$ should make the scores for both equal.

Using these 2 rules and setting $\forall k (w_{k_t} = 1)$, we get

$$w_{k_h} = \frac{C_t}{1 - e^{-\frac{\pi^2}{32}}} \qquad \text{and,} w_{k_d} = \frac{C_t}{1 - e^{-\frac{C_d^2}{2\sigma^2}}}$$

C_t and C_d are constants that have an explicit meaning as defined in the rules above and can be given values based on preference. For the experiments in this paper, we use C_t =100s and C_d =500m.

3.2 Relay State

In the relay state, the agent does not concern itself with the search operation and dedicates itself to relaying packets between GSs. When it comes to deciding on the next action, the agent only scores GS cells (those that are members of PL), using the formula in Equation 2. As a result, the agent picks the GS that obtains the highest score and heads towards it. If an agent k were to meet any other agent m in the RL or HB state (i.e. $m \in N_{RL,HB}(k)$), it immediately resets the $t_{k_{ij}}$ values in VM_k for the GS cells that are closer to the neighboring agent and recomputes its action. As a result, the following proposition holds true.

$$\forall (i,j) \in \text{PL} \left(\exists m \left(m \in N_{RL,HB}(k) \wedge d_{m_{ij}} < d_{k_{ij}} \right) \to t_{k_{ij}} = 0 \right) \tag{7}$$

The above rule is applied because the agent that is closer to a given GS should be in charge of delivering packets to that GS, and there is no point in sending more than one agent, moving together, towards the same GS. As an effect of this rule, the emergent behavior is the latency minimizing chain-relay architecture in the case of 2 GSs, illustrated in Figure 2. The chain-relay architecture as a fixed trajectory solution

Figure 2: Chain-relay architecture

is studied in detail in [1]. In the case of multiple GSs, we believe the emergent behavior would hold the same latency minimizing quality. The effectiveness is studied empirically in section 4.

3.3 Hybrid Search and Relay State

In the hybrid search and relay state, the agent performs the search operation as well as relaying of packets between GSs. Every time the agent needs to compute its next action, it scores all cells in the entire grid using the formula in Equation 2. In other words, it performs all the steps laid out in section 3.1. However, when computing scores for GS cells, the first term corresponding to elapsed time is given a lot more importance. In particular, $w_{k_t} t_{k_{ij}}$ becomes $w_{k_t} (t_{k_{ij}})^2$ so as to represent the higher visit frequency requirement of GS cells. When a GS cell gets chosen based on highest score, the agent implicitly works on relaying packets. In a similar manner to the RL state, whenever the agent gets within communication range of another agent in RL or HB state, the rule in Equation 7 is applied.

3.4 Proxy State

In the proxy state, the agent moves in a circular motion with minimum turn radius over the GS for which it acts as proxy. Having a proxy for every GS is essential in order to maintain the DTN protocol implementation on the agents, transparent to the GS. Equipment held by survivors and rescuers would in most likelihood use standard IP networking protocols. IP is not designed to be delay tolerant and lacks the key features of a DTN protocol: buffering, and opportunistic forwarding [3]. IP would simply drop all packets if no route existed to the destination. The proxy agent is used to stop IP from doing that by letting the GS know that a route exists. On receiving packets from the GS, the PR agent would only need to buffer all received packets and forward them when an RL or HB agent comes by. As a result of using proxy agents, the service time, s_g in Equation 1, is maximised for each GS g that has a proxy. The other task of the agent in PR state is to keep track of the GS it is in charge of. While flying in circles, the agent tries to maintain connectivity with the GS at all times. It maintains an estimate of the GS's position and circles around this point. Therefore, if it ever discovers that a part of the circle is beyond the range of the GS, it updates the position estimate by moving it a small distance directly away from the arc that fell out of range. The proxy agent for a given GS has complete authority over the position information for that GS. It is the only agent allowed to make changes to the position estimate for that particular GS, and when the estimate changes, the proxy agent informs the change to all agents that pass by.

Every proxy agent also maintains the average visit frequency for the GS over the past T_W time units. If this average visit frequency drops below a threshold, $\nu + \Delta\nu$, the agent decides to recruit an RL agent. Once the decision to recruit is made, the first agent that comes into contact with the PR agent is recruited as an RL agent. If recruitment is unsuccessful even after $T_{recruit}$ time units, the PR agent decides to personally forward the packets to the next closest known proxy agent (or base station) and informs the other agent to recruit an RL agent. The PR agent then returns to its corresponding GS and waits. If at any point the average visit frequency goes above $\nu + \Delta\nu$, the PR agent decides to dismiss an RL agent. The minimum time period between two consecutive recruitments or dismissals is T_W.

Since GS's can be intermittent, proxy agents continue circling for a fixed period of time, T_C, when the GS disappears. If the agent does not receive any wireless signal from the GS within T_C, the agent considers the possibility that the GS might have moved. It then begins spiraling outwards up to a distance beyond which the GS could not have traveled given a maximum speed of 10ms^{-1}. If the GS is still not found, the agent assumes the GS is lost and switches its op-

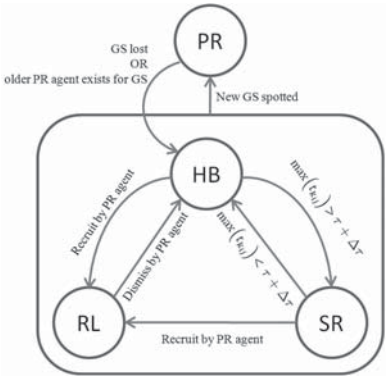

Figure 3: State Diagram

erational state to HB. It then spreads the information about the missing GS to other agents, thus causing them to delete the corresponding entry in their PLs.

3.5 State transitions

The state transitions are based on the state diagram shown in Figure 3. All agents start out in the HB state, which is equivalent to the SR state when no GSs have been found. An agent in the HB state decides to switch to the SR state, if any of the cells in VM_k has a $t_{k_{ij}}$ value of more than $\tau+\Delta\tau$. A high value on the VM means the search is slow and the average visit frequency is low, thus requiring a more dedicated search effort. Once the dedicated search manages to increase the average visit frequency, the SR agent should be able to switch back to HB. The switch from SR to HB takes place when the highest value on the VM becomes lower than $\tau+\Delta\tau$. The transition to the PR state can take place from any other state. When an agent spots a GS that does not already have a proxy agent, the agent immediately switches to the PR state and becomes in charge of that GS. If an agent mistakenly became the proxy agent of a GS with an existing proxy agent, the newer one reverts to the HB state and gives way to the original proxy for that GS. Otherwise, an agent in the PR state switches to the HB state only if the GS has been deemed lost as per section 3.4. The only way any agent can enter the RL state, is being recruited by a proxy agent. The recruitment process has been discussed in section 3.4. The transition from the RL state to the HB state happens only when the RL agent gets dismissed by a PR agent.

3.5.1 Adaptive state transition

The thresholds for state transitions are represented as $\tau+\Delta\tau$ and $\nu+\Delta\nu$ so that τ and ν can be kept constant while updating $\Delta\tau$ and $\Delta\nu$ to manipulate the thresholds. These thresholds indirectly determine the ratio of number of agents in the RL, HB and SR states. The ratios in turn affect the value of Q that we try to maximize in Equation 1. We know that s_{avg} is maximized as a side-effect of having proxy agents. As a result the two factors that are variable are l_{avg}, and f_{avg}. l_{avg} can be reduced by increasing the number of RL agents. However, that would adversely affect the search operation and reduce f_{avg}. The key is to balance them out so as to maximize Q. However, latency is highly dependent on the relative positions of the GSs (possibly mobile), and average visit frequency is dependent on the number of agents on field, both of which are not known for certain and are dynamic. In other words, the values of $\tau+\Delta\tau$ and $\nu+\Delta\nu$ also need to be dynamic so as to update the ratios of agents in different states, appropriately.

The correct way would be to evaluate Q from time to time and adapt the threshold values. However, in order to obtain values for l_{avg} and f_{avg}, global knowledge would be necessary. To overcome this problem, we propose the use of estimates for Q derived at each agent. The estimates can then be passed on to the base station if and when the agent comes in contact with the base. Using multiple estimates, the base station can make an informed decision as to how to update the threshold values. The updated values can be disseminated through agents that pass by. To be able to produce an estimate for Q, the agents need to perform 2 additional tasks:

1. Maintain a set TS_{ij} for each cell in the VM that holds timestamps of all the times when the value in the cell drops. Any element in the set that has a timestamp earlier than T_W time units prior to current time is discarded. $\frac{|\text{TS}_{ij}|}{T_W}$ then gives the the visit frequency for cell (i,j) over the last T_W units.

2. Update the timing information on the DTN protocol header (this field is assumed on the header, but even otherwise it is a single integer that is added to the header) by adding the duration for which the agent held the packet.

As a result, every agent would have its own version of f_{avg}. The agent that delivers a packet to the destination would have latency information for that packet. Using the latency information for all packets delivered in the last T_W time units, the agent can generate an estimate for l_{avg}. If and when an agent k passes by the base station, it delivers $\hat{f}_{avg_k}$ and $\hat{l}_{avg_k}$, which are estimates by agent k for f_{avg} and l_{avg} in Equation 1. An agent never modifies its trajectory with the aim of delivering these values to the base, because it is of lower importance than other operations and there is bound to be some agent that delivers packets to the base station that can provide its estimate. Finally, the two values can be used by the base to generate $\hat{Q}_k$, an estimate for Q. The base uses a reference value for the product of latency and average visit frequency. The reference value, λ, is calculated by the base station using its current knowledge of GS positions. λ is given by

$$\lambda = \frac{\sum_{g1,g2\in G, g1\neq g2} d_{g1\to g2}}{MN} \tag{8}$$

The idea is that l_{avg} should in general be proportional to the sum of distances between any pair of ground stations, and inversely proportional to UAV speed and number of UAVs if all agents were involved in relaying packets. Similarly, f_{avg} should be inversely proportional to area of the grid and directly proportional to UAV speed and number of UAVs if all agents were involved in the search operation. The product of the two should cancel out UAV speed and number of UAVs to give Equation 8. This is a combination of 2 individually optimal cases for latency and visit frequency. So the product is a reference value for the situation where equal number of agents are involved in each of search and relay operations. The base can then compute $\hat{f}_{avg}\hat{l}_{avg}$ to get an idea of relative distribution of SR and RL agents. This information as

well as the trend in Q can be used to obtain the update rule for $\Delta\tau$ and $\Delta\nu$ as follows:

$$\Delta\tau(t) = \frac{\tau}{\lambda}\left(\hat{f}_{avg}\hat{l}_{avg} - \lambda\right) + \Delta\tau(t-1)\text{slope}(\hat{Q}) \quad (9)$$

$$\Delta\nu(t) = \frac{\nu}{\lambda}\left(\hat{f}_{avg}\hat{l}_{avg} - \lambda\right) + \Delta\nu(t-1)\text{slope}(\hat{Q}) \quad (10)$$

Updates for both τ and ν are derived similarly. In Equation 9, the new $\Delta\tau$ depends on the previous $\Delta\tau$, as well as the deviation in the fl product from the reference value (scaled to the same order as τ), and finally the slope of the regression line through the last 20 estimates of Q. slope(Q) determines whether to switch the sign of $\Delta\tau$.

3.6 Belief Information Exchange

Belief information, i.e the VM and PL are exchanged when any two agents come within communication range of each other. It is important for every agent to know the locations of all GSs. This necessitates the full exchange of PLs. When an agent receives a neighbor's PL, it simply merges its own PL with the received PL

$$PL_k = \bigcup_{i \in (k \cup N(k))} PL_i$$

As for the VM, a full exchange would mean that each message would contain MN number of integer values, which would require a high bandwidth. In fact the bandwidth issue is a prominent one in the general cooperative target search problem. People have tried to address it by suggesting the exchange of only recently updated cell information [4]. This very often turns out to be only those cells that lie in the recent flight path of the sender. Since any cell can be on the list, the sender would also need to include co-ordinate information for each cell whose information is sent, thus tripling the number of bytes required to represent the information for 1 cell. Some others have suggested the exchange of last known positions of other agents along with their destinations [13, 14]. However, this would require each agent to maintain the history for all other agents. Moreover, if the destination of an agent is chosen as a relatively close point to current location, as in our case, there is very little information embedded.

We propose the use of neural networks to estimate the values of grid cells given some intelligently chosen partial information. We utilize the general tendency of agents to move in straight lines and the minimum turn radius of UAVs to come up with a pattern of cells whose information would be sufficient to obtain a good estimate of the remaining cells. The pattern we propose is the one in Figure 4. The effect is that any straight path taken by agents would intersect with a shaded cell atleast once every 5 cells. Moreover, the minimum turn radius of the UAVs wouldn't allow them to fly in a circle without cutting across a shaded cell. In reality though, no agent in a non-PR state would fly in circles.

Essentially, this pattern would require only $\frac{3}{8}$th the information for the entire grid. However, the packet size would increase linearly with respect to grid area and that is quadratically with respect to a given side, if it is a square. To overcome this problem, we divide the grid into horizontal strips each 8 cells thick, as shown in Figure 4. Each strip has an index number starting from 0 for the first strip. When a neighboring agent sends a packet, it specifies the index number and a sequence of integers giving values corresponding

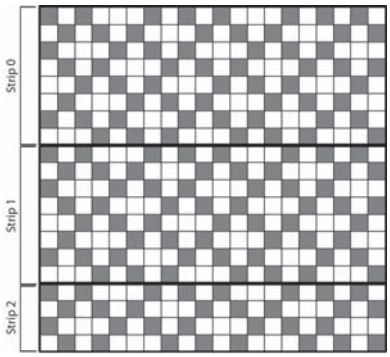

Figure 4: Pattern of cells chosen for exchange

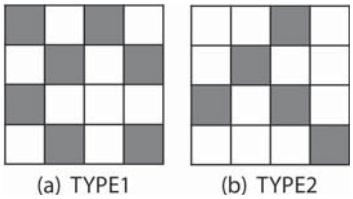

Figure 5: Types of blocks handled by each DCS

to shaded cells alone. The co-ordinate information is not required because the receiving agent knows exactly which cell each value corresponds to in the fixed pattern. Since the thickness of each strip is fixed, packet size would increase linearly with respect to one of the sides of the grid. The sending agent would first send the strip that contains the receiving agent. This would have information pertaining to the immediate environment of the receiving agent. Following this, the sending agent sends strips that are increasingly far away from the receiving agent on either side (above or below). The last strip sent would be the furthest from the receiving agent and would be of least immediate relevance.

Having received a packet, the agent estimates the values for the remaining cells using Dynamic Cell Structures (DCSs), which are basically radial basis function neural networks with lateral connections between neurons. The original DCS was proposed by Bruske and Sommer [2]. A detailed description of the modified, faster learning DCS used in this paper is available at [15]. The DCS is chosen because it is capable of differentiating between different situations by using lateral connections between neurons that are related. The hypothesis is that the DCS should be able to distinguish between different cases when an agent flies across the block horizontally or vertically, etc., and interpolate correctly. In order to avoid the curse of dimensionality, we extract 4×4 blocks from the grid, each of which serves as a data point. We identify 2 types of 4×4 blocks where each pattern in a type is only a rotated version of the corresponding block shown in Figure 5(a) and 5(b). To be more specific,

$$\forall i, j (i \text{ is even}) \wedge (j \text{ is even}) \rightarrow B_{ij} \in \text{TYPE1}$$

$$\forall i, j (i \text{ is odd}) \wedge (j \text{ is odd}) \rightarrow B_{ij} \in \text{TYPE2}$$

where B_{ij} is the block with top left corner at (i, j). Therefore, DCS1, after learning, takes a 7-tuple as its input corresponding to the shaded cells in Figure 5(a) and produces a 9-tuple output corresponding to estimates of the unshaded

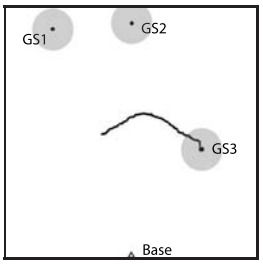

Figure 6: Positions of GS and path of mobile GS

cells. Similarly, DCS2 takes a 5-tuple as its input and produces a 11-tuple output. The data to train the two DCSs is obtained by running simulations with full communications (full exchange of VMs). From one snapshot of the 20×20 VM grid on one agent, 73 data points are produced for DCS1 and 72 data points are produced for DCS2. The simulation is run for 30 mins to generate millions of data points to train the DCS. Subsequent to supervised learning offline, DCS1 (93 neurons, 7% estimation error) and DCS2 (129 neurons, 7% estimation error) are used on every agent to estimate missing cells in information received from neighbors. For all experiments conducted in this paper, the DCS-based belief exchange mechanism is used.

4. SIMULATION AND RESULTS

The cooperative control architecture thus presented is implemented and tested using a combination of 2 simulators: X-Plane 8.64 for realistic UAV control and Qualnet 4.5 for realistic communications. Any communication between 2 UAVs goes through Qualnet, which determines whether the wireless transmission was a success based on its communication model. UAVs in this simulation have a maximum speed of 25ms^{-1}. The controller for the UAV is implemented using proportional integral derivative (PID) components and nonlinear dynamic inversion (NDI) components in order to achieve accurate waypoint navigation. The decisions taken by the agent are translated to target waypoints for the controller. A set of controlled experiments are conducted to observe change in behavior and performance of the cooperative control architecture with change in number of UAVs. A 2km×2km disaster area is used for all experiments. The agents start out at random positions on the field with no prior knowledge of ground stations except for the base station. Three GSs are used of which 1 is stationary and continuously emitting, while the other 2 are intermittent with one being mobile and the other, stationary. The initial positions of all GSs and the path for the mobile GS is as shown in Figure 6. The mobile GS stops emitting wireless signals for very short durations. GS2 on the other hand, only appears after 100s through the simulation and disappears again at 400s, before reappearing at 600s. Every GS generates packet streams of 10 packets/second towards every other GS, including the base station. Every experiment is run for 15 minutes and repeated 5 times to ensure no anomalous behavior.

The parameter varied between each experiment is number of agents. We start with 5 agents so that there are atleast 2 agents remaining when 3 of them become proxy agents. We increase the number of UAVs until 10. Figure 7(a) plots l_{avg} over time for one sample run of each experiment. We observe

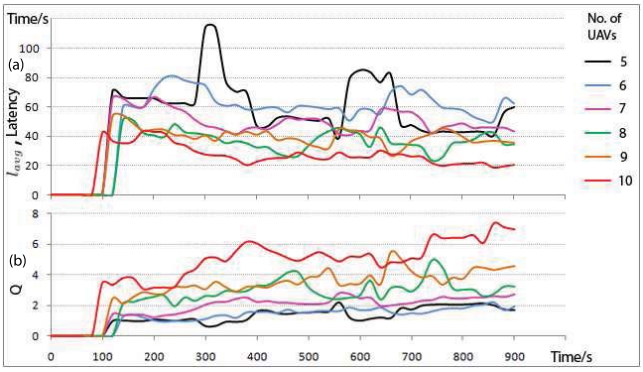

Figure 7: (a) l_{avg}, average latency of packets delivered in the last T_W time units; (b) Q based on Equation 1

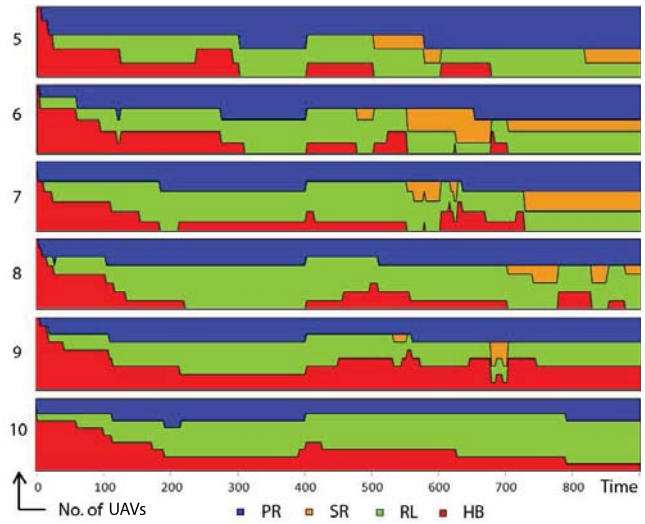

Figure 8: Distribution of planes in the 4 states through time for each experiment

the general tendency of latency to reduce with increase in number of agents. The interesting behavior though is the sudden spikes in latency at times when a new GS is found. This can be attributed to the fact that the new PR agent would have begun buffering packets from the new GS, but stayed unable to forward them until another agent came in contact with this PR agent. However, the adaptive state transition mechanism ensures that RL agents are introduced to minimize this latency. Within each experiment, we also observe that latency generally decreases. This is a direct consequence of trying to maximize Q in Equation 1, which is plotted in Figure 7(b) and shown to be generally increasing in value, albeit non-monotonically.

In Figure 8, we observe the relative distribution of agents in different states through time. We see how SR agents slowly become unnecessary with increase in number of agents because HB agents can already provide a satisfactory search effort. The consistently changing SR:HB:RL ratios we observe can be attributed to the effective adaptive state transition mechanism that modulates $\Delta\tau$ and $\Delta\nu$ to maintain a positive slope for Q.

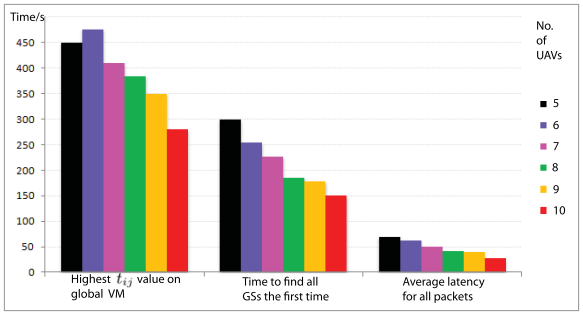

Figure 9: Global performance metrics (averaged over multiple runs of each experiment)

The values of maximum t_{ij} from the global version of the VM, are plotted in Figure 9. The longest any cell had to wait for a re-visit is very low for all the experiments considering 3 of the agents would have been in PR state. The observation that actually goes to show the effectiveness of the search method laid out in section 3.1, is the time taken to find all GSs (shown in Figure 9). The spread of a search needs to be wide with minimal overlaps to have a low find time. The results here go to show that the scoring mechanism in Equation 2 achieves exactly this. Finally, a relatively low average latency implies that the emergent network architecture is a latency minimizing one. The time taken for a UAV to fly from one GS to another is about 80s given the 25ms^{-1} speed of UAVs. To achieve a latency of as low as 20s on average (as shown in Figure 9), the distance a packet could have traveled on a UAV is quarter the distance between the GSs. The remaining distance would have had be covered by wireless transmission. Using the set of behaviors described for an agent, it is possible to determine that the agents would converge to a chain-relay architecture for 2 points on the ground. However, it is hard to determine analytically whether this latency minimizing behavior is extended to scenarios with more than 2 GSs. The experiments have proven that the mechanism adopted by the HB agents and RL agents, actually manages to keep the latency reasonably low.

5. CONCLUSION

In this paper, we have presented a novel cooperative control mechanism to coordinate a swarm of UAVs and establish a wireless communications backbone connecting multiple ground stations. In particular, 4 states of operation are introduced with an adaptive state transition mechanism. The adaptive update rule modifies the behavior of the swarm based on the current state, so as to minimize packet latency and maximize cell visit frequency. The search operation is designed in such a way that the same data structures can be used for the relay operation as well, thus enabling a hybrid search and relay state. We also take into consideration the bandwidth limitations of wireless links and present a novel solution to acquire fairly accurate information from neighboring agents despite using little bandwidth (specifically $\frac{3}{8}$th). To this end, a DCS-based state estimation procedure is proposed that utilizes knowledge of UAV dynamics restrictions and the general behavior of an agent. Empirical results have shown that the proposed mechanism is not only able to perform both the search and relay operation efficiently but also adapt to changing situations such as addition or loss of ground stations. The next step in this project shall involve implementation of the proposed mechanism alongside the set of MP2028 autopilots that we use to fly our Hangar 9 Alpha 60 UAVs. Further work on the algorithm front would involve exploring the idea of making each agent a reconfigurable formation of UAVs.

6. REFERENCES

[1] T. Brown and D. Henkel. On controlled node mobility in delay-tolerant networks of unmanned aerial vehicles. In *Int. Symp. Advanced Radio Tech.*, 2006.

[2] J. Bruske and G. Sommer. Dynamic cell structure learns perfectly topology preserving map. *Neural Comput.*, 7(4):845–865, 1995.

[3] V. Cerf *et al*. Delay-tolerant networking architecture. RFC 4838, Apr. 2007.

[4] P. DeLima and D. Pack. Maximizing search coverage using future path projection for cooperative multiple UAVs with limited communication ranges. In *Optimization and Cooperative Control Strategies*, pages 103–117. Springer, 2009.

[5] E. W. Frew and T. X. Brown. Networking issues for small unmanned aircraft systems. *Journal Intell. Robotics Syst.*, 54(1-3):21–37, 2009.

[6] A. Hafslund, T. T. Hoang, and O. Kure. Push-to-talk applications in mobile ad hoc networks. In *IEEE Vehicular Technology Conf..*, pages 2410–2414, 2005.

[7] D. Hague, H. T. Kung, and B. Suter. Field experimentation of cots-based UAV networking. In *Military Comm. Conf..*, pages 1–7, 2006.

[8] S. Hauert, J.-C. Zufferey, and D. Floreano. Evolved swarming without positioning information: an application in aerial communication relay. *Auton. Robots*, 26(1):21–32, 2009.

[9] I. Iridium Communications. Iridium, 2009. `http://www.iridium.com/`.

[10] Y. Z. Lee. *Scalable ad-hoc routing: design and implementation*. PhD thesis, 2008. Adv.-Gerla, Mario.

[11] S. Masaki and C. Wataru. Helicopter satellite communication system for disaster control operations ensuring prompt communications. *Journal of the IEICE*, 89(9):806–810.

[12] J. Ousingsawat and M. G. Earl. Modified lawn-mower search pattern for areas comprised of weighted regions. In *American Control Conf.*, pages 918–923, 2007.

[13] D. J. Pack, P. DeLima, G. J. Toussaint, and G. York. Cooperative control of UAVs for localization of intermittently emitting mobile targets. *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics.*, 39(4):959–970, Aug. 2009.

[14] P. Scerri, T. Von Gonten, G. Fudge, S. Owens, and K. Sycara. Transitioning multiagent technology to uav applications. In *Proc. AAMAS'08*, pages 89–96, 2008.

[15] A. Sivakumar, T. S. Phang, and C. K. Y. Tan. Stability augmentation for crosswind landings using dynamic cell structures. In *AIAA Guidance, Navigation and Control Conf. GNC'08*, Aug. 2008.

[16] A. Sivakumar, T.-S. Phang, C. K. Y. Tan, and W. K. G. Seah. Robust airborne wireless backbone using low-cost UAVs and commodity wifi technology. In *Int. Conf. ITS Telecomm.*, pages 373–378, 2008.