2018 IEEE International Conference on Robotics and Automation (ICRA)
May 21-25, 2018, Brisbane, Australia

Data Ferrying with Swarming UAS in Tactical Defence Networks

Robert Hunjet,[1] Bradley Fraser, Thomas Stevens,[2] Laura Hodges,[3] Karina Mayen, Jan Carlo Barca,[4]
Madeleine Cochrane, Ricardo Cannizzaro, and Jennifer L. Palmer[3]

***Abstract*— In this paper we categorise swarming into four classes, depending on the manner in which swarm members communicate. We identify two of these classes as ready candidates for the provision of communications within tactical defence networks in which radio-frequency communications are highly contested or denied. We demonstrate the feasibility of a swarm-robotics approach to data ferrying from both of these classes using simulation, emulation, and physical swarm robotic platforms within indoor flight facilities. The results show strong alignment between data dissemination capabilities of the simulated and physical systems; we envisage these techniques providing communications between not only troops, but also other swarm robotic platforms, thereby enabling swarm robotics applications and human-swarm interaction within harsh communications environments.**

I. Introduction

Within the tactical defence environment, communications capability is key to maintaining situational awareness and enabling timely decision-making. Given that tactical activity may take place at large distances from forward operating bases or within urban environments, the use of high-speed line-of-sight (LOS) communication bearers may be infeasible. Over large distances, beyond-LOS bearers such as satellites are usable; however, these provide attractive targets for an intelligent adversary and may not be available. The urban canyon presents its own challenges, with shadowing and radio-frequency (RF) interference from ubiquitous wireless communications systems [1]. In either case, even when tactical operations are within LOS communications range, maintaining communications can be problematic.

A complementary approach to communications is needed to ensure warfighters are provided with the information required for their mission. One such approach is data ferrying [2] which relies on autonomous, unmanned aircraft systems (UAS) ('drones'). However, in a denied-communication environment, data-ferry coordination can be challenging. In such cases, fully autonomous UAS control is required (UAS operate independently, without a control channel) and inter-UAS coordination must be decentralised. A swarm-based approach to UAS control can achieve this [3]. The intent of this paper is to show that the physical realisation of such systems is feasible with performance aligned to that obtained in simulation, albeit slightly degraded by real-world effects.

The paper is organised as follows, Section II categorises swarming into four classes; Sections III and IV present simulation, emulation, and physical realisation of two data-ferrying behaviours aligned with swarming classes appropriate for use in contested- or denied-communication environments. Section V presents conclusions and future work.

II. Background and Behavioural Characterisation

Information exchange between swarm members is required to create and maintain swarming. Without it, the swarm will stagnate, unless predefined rules are in place as a contingency. Below, we categorise swarms into four classes, based on the manner of intra-swarm communication.

A. Explicit Communication

Swarms that communicate explicitly transfer information to neighboring nodes (other swarm members) directly [4]. Explicit communication facilitates efficient information dissemination and is therefore often regarded as the most desirable form of communication. In contested-RF environments, explicit communication can be realised via direct information ferrying and delay-tolerant network-routing principles [5]. An alternative is to use cognitive radios, which exploit a full suite of anti-jamming mechanisms. Although such radios have been used by the US Air Force to operate small UAS [6], sophisticated electronic attacks ('jamming') may still cripple such systems. RF jamming may be overcome by switching the communication mode altogether. Alternatives include optical [7] and infrared systems.

B. Implicit Communication

Implicit communication is more indirect, but also more robust and generally harder to jam than explicit mechanisms [4]. However, it does not guarantee message delivery, and feedback is required to increase reliability. In implicit schemes, robots may act on traces of information deposited in the environment. Such traces can be realised via physical data caches or messages left behind for other robots to discover. Stigmergy is a well-known form of implicit communication [8]–[10], with environmental markings and pheromones being two approaches. Environmental markings can be produced via QR codes [11]–[13], colour tags, or paint blotches that strengthen as robots add to them. A drawback with these solutions is that signals do not weaken over time, and thus adequate security mechanisms to prevent

[1]Robert Hunjet is with Land Division, Defence Science and Technology (DST) Group, Edinburgh, SA, Australia.

[2]Bradley Fraser and Thomas Stevens are with Cyber and Electronic Warfare Division, DST Group, Edinburgh, SA, Australia.

[3]Laura Hodges, Madeleine Cochrane, Ricardo Cannizzaro, and Jennifer L. Palmer are with Aerospace Division, DST Group, Fishermans Bend, VIC, Australia. {firstname.lastname}@dst.defence.gov.au

[4]Karina Mayen and Jan Carlo Barca are with the Swarm Robotics Laboratory, Monash University, Clayton, VIC, Australia. {karina.mayengonzalez, jan.barca}@monash.edu

spoofing are harder to implement. In the natural world, pheromones, chemical signals secreted by insects and other animals, accumulate and evaporate in a manner that has been observed to result in complex global behaviour. Mechanical devices capable of altering their state to convey information and small mobile robots that operate as members of a swarm may practically emulate biological pheromones. Ideally a low/high technological mix should be employed to increase robustness against sophisticated adversaries capable of manipulating or disrupting pheromone-based communications.

C. Passive Action Recognition

It may be argued that passive action recognition is the most robust means for transferring data, as such mechanisms do not rely on a communication medium or environmental configuration [14]. Examples include leader–follower [15], [16] and waggle-dance approaches [4], which can convey complex information through sequences of movements [17]. Communication mechanisms inspired by waggle dancing have produced promising results with UAS [4]. For these methods to be successful, at least one swarm member in a pair must be capable of sensing the other with sufficient resolution to receive transmitted messages. Passive action recognition can therefore fail under extreme conditions, e.g. where smoke or water vapour obstructs vision. Communications reliant on passive action recognition are also limited by low data rates.

D. No Communication

Swarming cannot be sustained without intra-swarm information exchange, but it can be prolonged if robots are equipped with a commander's intent [18] or with a predefined sequence of actions. These strategies can take the form of an overall goal or objectives to be attained. Nodes aware of the intent can then independently labour toward the objectives when communication is fully denied.

E. Implications for Data Ferrying

The use of explicit communication to maintain a swarm of UAS may not be appropriate in an environment in which the communication channel is severely degraded. Because data ferrying may be conducted over large distances or in environments with visual obscuration, passive action recognition may be infeasible. In Sections III and IV, data-ferrying approaches relying on swarming UAS with *implicit communication* and *no communication*, respectively, are presented.

III. Stigmergic Data Ferrying

This section includes descriptions of simulation, emulation and physical experimentation with a stigmergic approach to data ferrying, an application of the *implicit-communications* class of swarming.

A. Simulation

The Java-based, discrete-event simulator MASON [19] was used to simulate a pheromone-based data-ferrying algorithm [20] in which it is assumed that ferries have *a-priori* knowledge of ground node ('user') locations and traverse the perimeter of the polygon defined by these locations in the same direction. This is an example of implicit communication in which user nodes continually broadcast a digital pheromone that decays with time. Upon a user—ferry interaction, the ferry (a UAS) sets its speed according to the residual pheromone level, i.e.,

$$s = s_{\min} + (s_{\max} - s_{\min})[1 - p(t_{\text{user}})], \tag{1}$$

where s, $s_{\min}$, and $s_{\max}$ are the ferry's speed and its minimum and maximum allowable speeds, respectively, in distance units per discrete time step; while $p(t_{\text{user}})$ is the unitless, time-dependent pheromone value on the interval $[0, 1]$. The pheromone decayed linearly with t_{user}, the time since the user was last visited by a ferry, according to

$$p(t_{\text{user}}) = \begin{cases} 1 - t_{\text{user}}/T & \text{if } t_{\text{user}} \leq T, \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

where T is the time for the pheromone to decay to zero.

After an interaction, the user node's pheromone signal was reset to the maximum value. Logically, if a ferry contacted a user with a high pheromone level, it was deduced that another ferry (or the one currently within range of the user) serviced that user recently. This caused the ferry to slow. Similarly, a low pheromone value caused the ferry to speed up. Ferries moved from user to user; over time, these simple rules resulted in an equal spacing of ferries around the perimeter of the polygon formed by the user nodes. A communications model was constructed so that direct communication between user nodes could not occur and a ferry had to be co-located with a user node to exchange signals with that user.

Consider the case in which user nodes are positioned on the vertices of a square with perimeter $P = 4L$, where $L = 6$ units, with the parameters for the algorithm listed in Table I. These values closely match the physical experiment described in Section III-C, ensuring:

1) timely convergence of ferry speed to a steady state [3];
2) a sufficiently large variation in ferry speed so that the emergent behaviour of the system was observable; and
3) safe operation of the experiment. It should be noted that the parameter selection for this dynamical system affects whether the system converges to a steady state or responds in a periodic or chaotic manner. For analysis of this bifurcation, see [3].

Simulations were run for 900 time steps (corresponding to 15 min in the experiment) with the final distance separating ferries and their final speed(s) recorded. During simulation, the number of time steps taken to distribute data packets (generated by each user at every time step) to all other user nodes, the *dissemination time*, was recorded. The mean dissemination time during the last 10% of the simulation was computed to capture its value at (or near) convergence. The distance between ferries at convergence is P/N, where N is the number of ferries. Thus the final speed of all ferries [3]

TABLE I

ALGORITHMIC PARAMETERS AND THE MEAN (μ), STANDARD DEVIATION (σ), AND CALCULATED (c) RESULTS FOR STIGMERGIC DATA-FERRYING SIMULATIONS CONDUCTED WITH FOUR USER NODES ON A 6- × 6-UNIT SQUARE.

No. of Ferries, N	Parameters			Separation Distance (units)			Final Speed (units/step)			Dissemination Time (steps)		
	$s_{\min}$ (units/step)	$s_{\max}$ (units/step)	T (steps)	μ	σ	c, P/N	μ	σ	c, Eqn. 3	μ	σ	c, Eqn. 4
1	0.1	0.5	24	N/A	N/A	N/A	0.50	0.00	0.5	60.49	13.81	60.00
2	0.1	0.5	45	12.00	0.01	12.00	0.38	0.01	0.38	64.08	9.30	63.09

is given by:

$$s_f^* = \begin{cases} \frac{s_{\min}+\sqrt{s_{\min}^2+4P(s_{\max}-s_{\min})/NT}}{2} & \text{if } \frac{P}{NT} \leq s_{\max} \\ s_{\max} & \text{otherwise.} \end{cases} \quad (3)$$

At convergence, the *expected* mean data-dissemination time for equally spaced ferries on a square is

$$\frac{3L + \frac{1}{2}(P/N)}{s_f^*}. \quad (4)$$

Table I shows results for two series of 1000-run simulations, conducted for single- and dual-ferry cases; and Fig. 1 shows the speed of the ferries during a run of the dual-ferry case. The results display several interesting features, described below.

1) The single-ferry scenario is a case in which $(P/N)/T > s_{\max}$, which would yield a final ferry speed of 0.68 units/step, if the computed value was not limited to $s_{\max}$. The standard deviation of the dissemination times obtained in the single-ferry simulations is largely due to users having to queue packets for greatly varying amounts of time before they are retrieved.
2) The final ferry speed and dissemination time for the dual-ferry case are in close agreement with the computed values, with convergence to a value of s_f that is less than $s_{\max}$ within 300 time steps.
3) The dual-ferry case achieved a slower dissemination time than the single-ferry case, due to the lower converged speed. This highlights the trade-off between using a single, faster-moving ferry, at the expense of UAS battery life, or two slower-moving ferries, at the expense of service delivery.

B. Emulation

Software emulation facilitated testing with a set of conditions scaled-up from those used in simulation. Each ground (user) and rotorcraft node was emulated with a Linux virtual machine. Radio communications were emulated on a common message bus between the virtual machines with 802.11 Wi-Fi propagating as per the log-normal shadowing model [21]. The rotorcraft was modelled with low-fidelity simulated physics. Its control and environmental sensing were implemented through software known as the Hardware Abstraction and Integration Layer (HAIL) [22] (described in Section III-C). This software enabled re-use of the algorithm executables when transitioning to physical testing. The geometric shape for the emulated test was four user nodes arranged in a 5×5km square. The algorithmic parameters shown in Table II were selected for fast, real-time convergence.

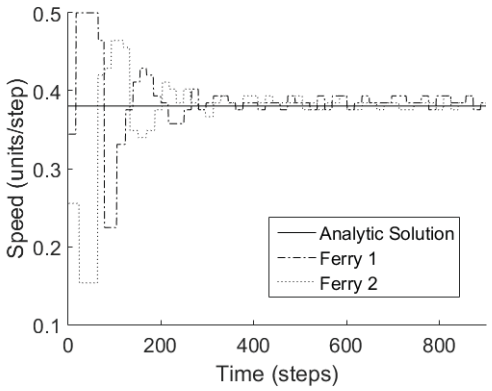

Fig. 1. The speed responses of two simulated ferries servicing four user nodes on a 6×6-unit square.

Ferries began in random initial positions near one of the user nodes, with their speeds set randomly between $s_{\min}$ and $s_{\max}$. RF-propagation values were configured so that communication was readily available with up to 50-m separation, fading rapidly beyond this. For each scenario, five emulations were run for 30 min each; and the final *service time*[1], the interval between ferry visits to a given ground node, was evaluated. Table II shows the mean results for five runs of each case. The emulated results are in close agreement with the analytical results. The differences are attributable to delays imposed by the aircraft model (e.g., finite acceleration) and Wi-Fi-connection times.

The time series in Fig. 2 show the speed of the ferries during a two-ferry experiment with 5×5-km square of user nodes. The red and orange traces show the desired speeds of the ferries as demanded by the algorithm, while the blue and green traces show their actual speeds, as reported by the emulated aircraft controller. Abrupt drops in a ferry's speed signify occasions when the rotorcraft reached a user node and changed direction. Over time, speeds converge to steady values that are identical for both ferries.

During this time series, ferry 1 was first to interact with the nearest user. Because user nodes began with a pheromone

[1]Service time is a measure closely related to ferry spacing; and, in the ideal case, its converged value is given by $P/N/s_f^*$.

TABLE II

ALGORITHMIC PARAMETERS AND THE MEAN (μ), STANDARD DEVIATION (σ), AND CALCULATED (c) RESULTS FOR STIGMERGIC DATA-FERRYING EMULATION CONDUCTED WITH FOUR USER NODES ON A 5×5-KM SQUARE.

No. of Ferries, N	Parameters			Final Speed (m/s)			Service Time (s)			Dissemination Time (s)		
	$s_{\min}$ (m/s)	$s_{\max}$ (m/s)	T (s)	μ	σ	c, Eqn. 3	μ	σ	c, $P/N/s_f^*$	μ	σ	c, Eqn. 4
2	57.7	97.5	130	92.9	0.8	91.3	115.5	3.5	109.6	233.3	34.7	219.2
3	47.3	91.5	271	65.0	0.3	64.2	108.0	1.9	103.8	301.3	42.4	285.4

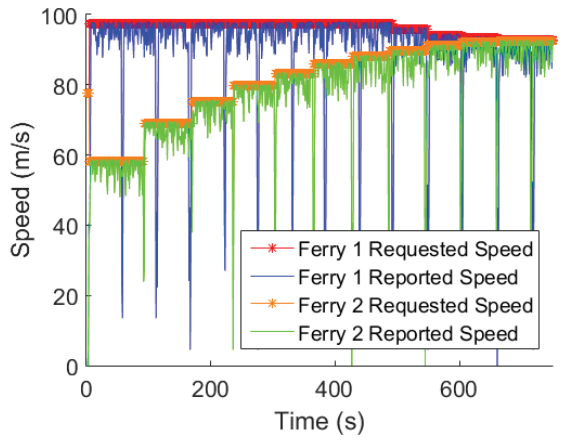

Fig. 2. The speed response of the rotorcraft ferries during an emulated experiment in which two ferries serviced four ground nodes arranged on a 5×5 km square.

level of zero, ferry 1 attained the highest possible speed. It maintained this speed for nine node visits, just over two circuits of the square. During this time, the pheromone set by the trailing ferry (2) had decayed to zero before ferry 1's next interaction. On the tenth visit, the pheromone set by ferry 2 had not yet decayed to zero. Ferry 1 thus began to slow. In a similar fashion, ferry 2 gradually its increased speed because it encountered progressively lower pheromone levels at each user node. These results demonstrate the resilience of the algorithm to timing perturbations resulting from software delays, such as algorithm timers, and from delays caused by physical modelling of the aircraft which imposed variation in acceleration when approaching and leaving nodes.

C. Physical Realisation

Real-world effects, e.g. aircraft acceleration and instability, can affect algorithmic performance. This section examines these effects though tests conducted with real quadcopters.

1) Platform, Testing Environment and Control: Testing was conducted in DST's Indoor Flight Laboratory, which is fitted with an Optitrack motion-capture system. The system consists of eight Prime 17W and forty Prime 13W cameras managed with OptiTrack's Motive software [23]. It tracks infrared LEDs arranged in distinctive patterns on each quadcopter; and the tracking data for each quadcopter is streamed via Wi-Fi to an on-board Odroid XU4 computer.

Each quadcopter weighed ∼1.5 kg and was 400 mm in diameter, with propellers in an X4 configuration. Each was equipped with a Pixhawk flight controller operating with PX4 V1.3.1 firmware. High-level control was managed by the on-board Odroid, which operated with the Robot Operating System (ROS) [24] running on Lubuntu 14.04.5. Our ROS-based control software managed the high-level operation and tasking of each quadcopter and accepted algorithmic input via HAIL [22]. A 5200-mA·h battery provided a maximum flight time of ∼10 min.

HAIL [22] provided an interface [25] between the ROS-based software and the stigmergic data-ferrying algorithm, relaying quadcopter status information from the ROS-based software to the algorithm. A HAIL mobility agent translated navigation directives of the algorithm (i.e. waypoint and velocity settings) into commands issued to the ROS-based software. A HAIL-ROS interface provided translation between Java and the ROS messaging bus and node-registration system. The interface, depicted in Fig. 3, was built utilising the ROSjava implementation [26].

HAIL ensured compliance with velocity and positional bounds and confirmed aircraft readiness before messages were translated and sent to the ROS-based software. The interface also provided quadcopter state information to the algorithm. Command-acceptance functions enabled reliable command delivery and notifications of command rejection, whilst also providing a 'safety-gate', only allowing the forwarding of algorithmic commands if "Autonomy Mode" was selected. This was enabled only once the quadcopter was airborne and ready to commence autonomous navigation.

HAIL's use as the hardware interface for the algorithm allowed seamless transition from emulation to physical implementation (the interface remained the same) necessitating only configuration changes to the HAIL software.

The ROS-based software provided a control interface for the quadcopters. This software ingests commands from a variety of sources in a variety of forms. In this case, input was taken from a command-line interface (as takeoff, altitude-locking, and landing commands) and the HAIL interface (waypoint commands). The software included a pseudo-geofence, to enforce safe operating limits, and a state machine, to ensure sensible acceptance and rejection of commands during a flight. It also included conversion protocols, for example, between the geo-referenced coordinate system used by HAIL and its own local coordinate system. The software applied safety and bounds checking to all commands and

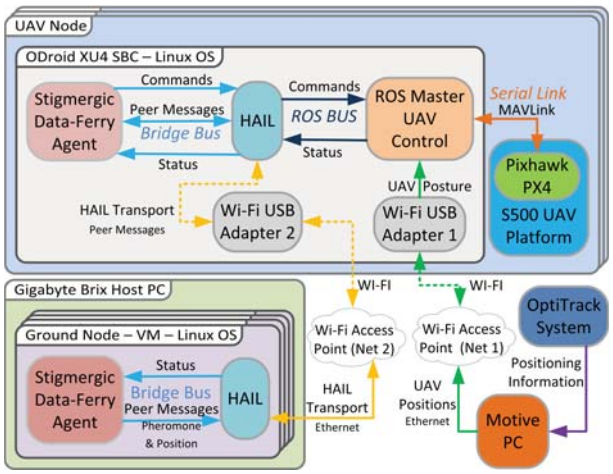

Fig. 3. Experimental setup utilising HAIL, ROS, and OptiTrack.

parameters, ensuring they were within the operating ranges. It then provided feedback to the external tasking software (HAIL) as to the acceptance of the request or the reason for rejection, optionally followed by periodic updates on its progress towards them. Finally, the software sent the final output commands to the Pixhawk, via MAVLink messages over a serial link.

During the tests, quadcopter speed was limited to 0.5 m/s to ensure safe operation. A pseudo-geo-fence bounded the flight space and prevented the quadcopters' moving outside the coverage of the motion-capture system or colliding with the walls of the laboratory. When multiple quadcopters were flown, they were separated in altitude by 2 m. A takeoff sequence, designed to minimise the effects of downwash on the lower quadcopter, was implemented as follows: the higher quadcopter was positioned closer to the first node and launched first via the ROS-based software. As it ascended to its flight altitude, the second ferry was launched. Once both were locked into their flight altitudes, HAIL was switched to Autonomy Mode and began controlling the ferries, commanding the first, then the second, to move towards the first waypoint. This sequence ensured a staggered arrival at the first waypoint and minimised interference between quadcopters.

During flight, the accuracy of each quadcopter's position was monitored with the Motive software. Each quadcopter also had an instance of QGroundControl [27] to monitor Pixhawk telemety data, and a remote pilot on standby, ready to take manual control in the event of unexpected behaviour.

The HAIL graphical user interface (GUI) showed the position, velocity, current waypoint and status of each quadcopter. The HAIL GUI also provided the ability to issue manual waypoint, velocity, and takeoff commands in addition to changing the HAIL Autonomy Mode, controlling the flow of commands from the algorithm to the quadcopter. The ROS command-line interface provided feedback on the high-level operation of the quadcopters, including state information and the processing of data received from HAIL. These interfaces,

TABLE III

MEAN (μ) SERVICE TIMES AND CONFIDENCE INTERVALS (95% CI) OBSERVED AT CONVERGENCE IN STIGMERGIC FERRYING EXPERIMENTS.

	Service Time (s)			
Ground-	Single-Ferry		Dual-Ferry	
node no.	μ	95% CI	μ	95% CI
1	61.0	2.3	35.8	1.5
2	61.0	1.5	35.5	2.5
3	61.2	1.4	36.2	1.7
4	60.8	2.0	35.0	1.8

and the remote pilots, provided monitoring and allowed human-on-the-loop control.

The ground nodes consisted of four Ubuntu Linux virtual machines running on a Gigabyte Brix PC. These were bridged to a single Ethernet network interface, attached to a Wi-Fi access point. Pheromone readings were accessed by the quadcopters through HAIL's multi-cast service using a second USB Wi-Fi communication payload, with the networks configured on non-conflicting channels. The ground nodes were placed on the vertices of a virtual rectangle 6.0 m $\times$ 5.4 m in size; and the communication range was artificially limited to 0.2 m in the horizontal plane, to mimic a sparse, disconnected network.

Finally, the Java-based, stigmergic-algorithm recorded system variables such as requested speeds and pheromone levels that could be accessed via secure shell or virtual network computing applications.

2) Results: The stigmergic data-ferrying algorithm was tested with a single quadcopter and with dual quadcopters using parameters comparable to those listed in Table I (i.e., $s_{\min} = 0.1$ m/s and $s_{\max} = 0.5$ m/s, with $T = 24$ and 45 s, respectively, for the single- and dual-ferry cases). The converged service time of each ground node was measured over five and six runs for the single- and dual-ferry cases, respectively. Experiments were conducted for the maximum flight time or until the algorithm converged (i.e. the service times for a given ground node changed by no more than 1 s over three successive ferry laps).

Table III shows the results for measured service time at convergence. The mean for each node and the 95% confidence interval (CI) around the mean are provided. Histograms of the service time (omitted due to space constraints) indicate an approximately normal distribution.

The service times calculated for the single- and dual-ferry experiments are 45.6 and 30.6 s, respectively. It can be seen that the measured values are considerably larger than these predictions. This is due to the dynamics of real flight: the perimeter traversal was predicted to be $[2 \cdot 6.0\,\text{m} + 2 \cdot 5.4\,\text{m}]/0.5\,\text{m/s} = 45.6\,\text{s}$.

Additional time taken in the experiment is attributable to time lost as the quadcopter decelerated on approach to each ground node and reaccelerated to the desired speed after an interaction.

Speed perturbations can be seen in Fig. 4, which shows

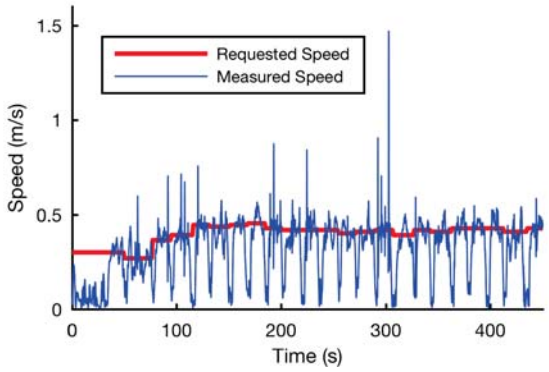

Fig. 4. The speed response of a quadcopter during a stigmergy-based data-ferrying experiment with two ferries servicing four ground nodes at the vertices of a 6.0- × 5.4-m rectangle.

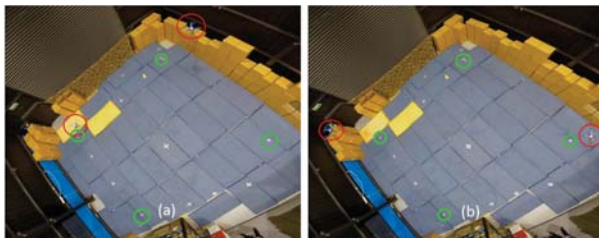

Fig. 5. Snapshots of a stigmergy-based dual-ferry experiment acquired (a) just after its start and (b) after convergence. The red circles highlight the quadcopters, and the green circles show the positions of the virtual ground nodes.

the speed commanded by the algorithm and the actual speed of one of the quadcopters during a dual-ferry experiment. Despite some perturbations, the algorithm converged, as evidenced by the small confidence intervals for service time in Table III. Similar algorithmic resilience was noted in the emulation results. Fig. 5 shows overhead views (captured with a ceiling-mounted GoPro camera) of (a) the beginning of the experiment, when both quadcopters are first on the perimeter of the rectangular flight area, and (b) the positions of the quadcopters near the end of the experiment, after the algorithm had reached convergence.

IV. Aggregation-Based Data Ferrying

In this section, simulation and physical experimentation with a node-aggregation-based approach to data ferrying are described. This method represents an application of the *no-communications* class of swarming. Rotorcraft assigned to specific user-nodes repeatedly meet (aggregate) at various points on a virtual ellipsoid to exchange information. The manner in which this was achieved is described below.

A. Simulation

Aggregation behaviours in a simulated swarm formation were achieved by assigning locally oriented potential fields [28] to each particle (swarm node), the characteristics of which are described within a reference system centred on the particle. These fields attract or repel particles via virtual forces determined by the absolute magnitude and direction of vectors representing the spatial relationship between a given particle, its neighbour(s), and its respective user-node. The sum of attractive and repulsive forces provides a resultant 'force' exerted by the potential fields of all the particles on the particle in question [29]. This requires the knowledge of particle locations and, as such, is applicable to environments in which this information can be ascertained, and data ferrying is still required, e.g., due to high background noise, or the requirement to maintain a low RF signature. To enable aggregation, the meeting point is introduced as a virtual attraction particle, exerting a force on two swarm nodes; at the same instant, the force of these swarm nodes' respective user nodes is set to zero, creating an imbalance and drawing the two swarm nodes towards the meeting point and, therefore, each other. Once the information exchange (data ferrying) is completed, the virtual attraction particle of the meeting point is removed, and the user-node forces return to their prior values, attracting the individual swarm nodes back to the vertices at their user nodes. At all times, the particles of the swarm exert a repulsive force on each other to avoid collisions. The swarm particles can be thought of as nodes, with controlled velocities and headings, moving in a potential field generated from a bivariate normal 'hill' [28], [30]:

$$f(x,y,z) = e^{-\kappa[(x-x_c)^2+(y-y_c)^2+(z-z_c)^2]}. \tag{5}$$

where the current location of a particle is (x, y, z), the centre of the neighbour is (x_c, y_c, z_c), and κ is a scaling parameter, equal to $0.001\,\mathrm{m}^{-2}$ in our case.

In our implementations (simulation and physical), the infinite tails of the attractive/repulsive potentials were cut-off to ensure collision avoidance by using the functions:

$$S_{\text{in}} = \epsilon \text{ and} \tag{6}$$

$$S_{\text{out}} = 1 - \epsilon, \tag{7}$$

where ϵ is a dimensionless control parameter that is arbitrarily small, but greater than zero, 0.0001 in our case.

The velocity vector for a node, n, $(v_{x,n}, v_{y,n}, v_{z,n})$, with inter-node collision avoidance was calculated using

$$\begin{bmatrix} v_{x,n} \\ v_{y,n} \\ v_{z,n} \end{bmatrix} = \sum_{n\prime=1}^{N} \left\{ (-S_{\text{in},n\prime} + S_{\text{out},n\prime}) \begin{bmatrix} d_{x,n\prime} \\ d_{y,n\prime} \\ d_{z,n\prime} \end{bmatrix} \right\}, \tag{8}$$

where N is the number of swarm nodes, $\{n\prime \in N : n\prime \neq n\}$. and $d_{x,n\prime}$, $d_{y,n\prime}$, and $d_{z,n\prime}$ are the partial derivatives of the bivariate normal equation (5). It is assumed that intra-swarm clocks are synchronised and that swarm nodes are capable of determining relative distances to their neighbours [31].

The coordinates for a meeting an aggregation point (rendezvous point) are defined by

$$\begin{aligned} x &= x_0 + ra\cos\theta\sin\varphi \\ y &= y_0 + rb\sin\theta\sin\varphi \\ z &= z_0 + rc\cos\varphi, \end{aligned} \tag{9}$$

where x_0, y_0, and z_0 are the centre coordinates of the ellipsoid, r is the virtual ellipsoid radius (the distance from

a swarm node to a point halfway between itself and its counterpart swarm node with which it is about to rendezvous); and a, b, and c are dimensionless scaling factors of the x, y, and z axes, respectively, of the ellipsoid. In our implementation, $a = 1.0$, $b = 0.5$, and $c = 0.2$. The variables θ and φ, where $-\pi/2 \leq \theta < \pi/2$ and $-\pi \leq \varphi < \pi$, can be interpreted as spherical coordinates. The values of θ, φ, and r are altered in a predetermined manner over time (without requiring information exchange) to prevent an adversary's predicting rendezvous location and time.

Simulations were conducted using a PC equipped with an Intel Core i7 processor and 8-GB RAM running 64-bit Ubuntu, ROS [32], Indigo 1.11.21, and Gazebo's Hector Quadrotor model. Rotorcraft flight was simulated at 10 Hz; and the simulated environment was modelled after an indoor testing facility at Monash University, which is described in Section IV-B. Four simulation configurations were created by initially placing the quadcopter models on the ground along the perimeter of a rectangle, which was incrementally scaled from 1.25×0.75 m to 2.50×1.50 m, 3.75×2.25 m, and 5.00×3.00 m. Each simulated quadcopter was then directed to the corresponding vertex of a rectangle located 1.5 m above the ground. Note, each vertex represents the location of a user node's particle, however the user (which is being serviced by the swarm) is notionally located directly below the vertex at ground level.

Data ferrying was performed by pairs of quadcopters rendezvousing on ellipsoids centered on edges connecting adjacent vertices. The major axis of each ellipsoid was defined by the corresponding edge length. Pairs of quadcopters sequentially rendezvoused until a particular data packet propagated throughout the swarm. Each simulation was repeated 100 times. The times taken to rendezvous (to meet at a location after departing from vertices) and to return (to leave a rendezvous point and return to the vertices) along the sides of the four rectangles are reported in Table IV. Rendezvous and return times along the long and short sides of a rectangle are abbreviated $RenTL$ and $RetTL$, respectively. The total time to disseminate data throughout the swarm is denoted by $TotTD$.

From the simulation results in Table IV one can observe:

1) It took longer for a quadcopter pair to return to a vertex than to rendezvous. This is because, to prevent collisions, larger separations were allowed between nodes at a rendezvous point than at a return point.
2) $\sim$30% more time is taken to rendezvous along the length of a rectangle than along its width. This is reasonable, given the aspect ratio of the rectangles.
3) Performance is reliable, given that the maximum standard deviation in rendezvous and return times was 0.06 s.
4) $TotTD$ increased by $\sim$14% with each size increment, indicating that the algorithm performed well with increasing formation size.

These results demonstrate that aggregation-based data ferrying enables data transfer in a simulated swarm setting.

Fig. 6. (a) Crazyflie equipped with a 3D-printed structure holding retro-reflective markers and (b) four quadcopters ferrying data.

B. Physical Realisation

Experiments were conducted to examine how the aggregation-based data-ferrying algorithm translates to the real world. A PC equipped with an Intel Core i7 processor and 8-GB RAM running 64-bit Ubuntu, 14.04 LTS, and ROS Indigo 1.11.21 acted as a base station, which executed the data-ferrying algorithm and transmitted high-level control commands to four Crazyflie 2.0 [33] quadcopters via two USB Crazyradios-PA [33]. Each quadcopter was 100 mm in diameter and 50 mm high, with a total weight of 45 g; and each was equipped with a lightweight, 3D-printed structure holding four retro-reflective markers arranged in a unique pattern. Two Crazyradios were physically separated by 5 m to reduce interference, and only two quadcopters were bound to each radio to prevent delays from multiplexing.

The Crazyflies were operated within a volume of $\sim 5 \times 3 \times 4$ m defined by fifteen OptiTrack Flex 13 motion-capture cameras [23]. Optitrack's Motive V1.8.0 software, running on a PC equipped with an Intel Core i5 processor, 8-GB RAM, and 64-bit, was used to monitor the position and orientation of each quadcopter. Position data was then transferred back to the base station via Ethernet.

The experiments, conducted with node arrangements, influence parameters (κ), and rectangle sizes identical to those used in the simulations, were repeated five times for each rectangle. Means and standard deviations of the results for rendezvous, return, and total dissemination times are presented in Table IV, where the nomenclature is as defined for the simulations. From these results, one can observe that:

1) $TotTD$ increased linearly with the size of the rectangle, at a rate of 5.9 s with each size increment. In contrast, in the simulations, $TotTD$ increased linearly, but at a slower rate (1.9 s with each size increment). This is attributable to nonidealities in the experiments.
2) The maximum standard deviation of rendezvous and return times is 0.61 s for the largest rectangle, which is an increase of 0.55 s, when contrasted with the simulations. This increase is also acceptable given that the algorithm ferried data successfully in all trials.

In summary, we conclude that the formulated data-ferrying mechanism translated well from simulation to the real world.

V. Conclusions and Future Work

This paper described a scheme for classifying swarming based on the communication requirement to maintain the

TABLE IV

MEAN (μ) AND STANDARD DEVIATION (σ) OF RENDEZVOUS, RETURN, AND TOTAL DATA-DISSEMINATION TIMES OBTAINED IN DATA-FERRYING SIMULATIONS AND EXPERIMENTS.

Rectangle size (m)	Simulation Results (s)										Experimental Results (s)									
	$RenTL$		$RetTL$		$RenTS$		$RetTS$		$ToTD$		$RenTL$		$RetTL$		$RenTS$		$RetTS$		$ToTD$	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
1.25×0.75	1.38	0.00	3.07	0.00	1.07	0.00	2.31	0.00	12.28	0.00	2.57	0.19	1.49	0.07	1.73	0.61	1.43	0.19	11.33	0.27
2.50×1.50	1.84	0.04	3.19	0.00	1.14	0.05	2.95	0.00	14.15	0.01	3.77	0.28	2.14	0.25	3.28	0.37	2.19	0.14	17.31	0.31
3.75×2.50	2.21	0.00	3.45	0.02	1.61	0.05	3.16	0.04	16.09	0.02	4.85	0.26	2.83	0.09	4.50	0.30	3.09	0.10	22.98	0.19
5.00×3.00	2.76	0.04	3.70	0.06	1.76	0.05	3.24	0.04	17.92	0.04	5.99	0.61	3.61	0.15	5.89	0.21	3.90	0.09	29.02	0.28

swarm. Two of these classes, *implicit communication* and *no communication*, were deemed appropriate for realising data-ferrying in contested to denied communication environments. A data-ferrying approach for each class was shown to be feasible in simulation and a physical incarnation. The stigmergic approach provided steady inter-node arrival times and the aggregation-based approach scaled well with formation size. Both approaches exhibited increased data dissemination times when translated from simulation to physical rotorcraft.

Implementation within indoor flight facilities at DST and Monash University's Swarm Robotics Laboratory showed the feasibility of swarm robotics-based approaches for communications through data ferrying. Future work will scale-up the experiments by implementing the schemes outdoors and increasing node numbers.

REFERENCES

[1] M. Elbanhawi, A. Mohamed, R. A. Clothier, J. L. Palmer, M. Simic, and S. Watkins, "Enabling technologies for autonomous MAV operations," *Prog. Aerosp. Sci.*, vol. 91, pp. 27–52, May 2017.

[2] K. Usbeck, M. Gillen, J. Loyall, A. Gronosky, J. Sterling, R. Kohler, R. Newkirk, and D. Canestrare, "Data ferrying to the tactical edge: A field experiment in exchanging mission plans and intelligence in austere environments," in *2014 IEEE Military Communications Conference*, Oct 2014, pp. 1311–1317.

[3] B. Fraser, R. Hunjet, and A. Coyle, "A swarm intelligent approach to data ferrying in sparse disconnected networks," *International Journal of Parallel, Emergent and Distributed Systems*, 2017 (in press).

[4] B. Das, M. S. Couceiro, and P. A. Vargas, "MRoCS: A new multi-robot communication system based on passive action recognition," *Rob. Auton. Syst.*, vol. 82, no. C, pp. 46–60, Aug. 2016.

[5] F. Warthman, "Delay and disruption-tolerant networks (DTNs)," DTN Research Group, Tech. Rep., 2015.

[6] C. Bostian and A. Young, "The application of cognitive radio to coordinated unmanned aerial vehicle (UAV)," 2011.

[7] J. Llorca, S. D. Milner, and C. C. Davis, "Mobility control for joint coverage-connectivity optimization in directional wireless backbone networks," in *MILCOM 2007 - IEEE Military Communications Conference*, Oct. 2007, pp. 1–7.

[8] R. Calvo, A. A. Constantino, and M. Figueiredo, "A multi-pheromone stigmergic distributed robot coordination strategy for fast surveillance task execution in unknown environments," in *International Joint Conference on Neural Networks (IJCNN)*, Jul. 2015, pp. 1–8.

[9] F. Ducatelle, G. A. Di Caro, and L. M. Gambardella, "Cooperative stigmergic navigation in a heterogeneous robotic swarm," in *From Animals to Animats 11*. Springer, Berlin, Heidelberg, 25 Aug. 2010, pp. 607–617.

[10] J. P. Hecker and M. E. Moses, "Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms," *Swarm Intell*, vol. 9, no. 1, pp. 43–70, 15 Feb. 2015.

[11] S. Alers, B. Ranjbar-Sahraei, S. May, K. Tuyls, and G. Weiss, "An experimental framework for exploiting vision in swarm robotics," in *ADAPTIVE 2013, The Fifth International Conference on Adaptive and Self-Adaptive Systems and Applications*, 2013, pp. 83–88.

[12] S. Alers, K. Tuyls, B. Ranjbar-Sahraei, D. Claes, and G. Weiss, "Insect-inspired robot coordination: foraging and coverage," *Artif. Life*, vol. 14, pp. 761–768, 2014.

[13] A. H. Ismail, S. J. Jamil, A. Hilmy Ismail, M. N. Ayob, and N. Abdul Rahim, "A comprehensive study of using 2D barcode for multi robot labelling and communication," *International Journal on Advanced Science, Engineering and Information Technology*, vol. 2, no. 1, pp. 80–84, 2012.

[14] M. J. Huber and E. H. Durfee, "Deciding when to commit to action during observation-based coordination," *ICMAS*, 1995.

[15] J. C. Barca, A. Sekercioglu, and A. Ford, "Controlling formations of robots with graph theory," in *Intelligent Autonomous Systems 12*. Springer, Berlin, Heidelberg, 2013, pp. 563–574.

[16] W. L. Seng, J. C. Barca, and Y. Ahmet Şekercioğlu, "Distributed formation control of networked mobile robots in environments with obstacles," *Robotica*, vol. 34, no. 6, pp. 1403–1415, Jun. 2016.

[17] T. D. Seeley, P. Kirk Visscher, and K. M. Passino, "Group decision making in honey bee swarms," *American Scientist*, vol. 94, no. 3, pp. 220–229, 2016.

[18] D. S. Alberts and R. E. Hayes, "Planning: complex endeavors," DTIC Document, Tech. Rep., 2007.

[19] S. Luke, C. Cioffi-Revilla, L. Panait, K. Sullivan, and G. Balan, "MASON: A multiagent simulation environment," *Simulation*, vol. 81, no. 7, pp. 517–527, 19 Jul. 2005.

[20] B. Fraser and R. Hunjet, "Data ferrying in tactical networks using swarm intelligence and stigmergic coordination," in *26th International Telecommunication Networks and Applications Conference (ITNAC)*, Dec. 2016, pp. 1–6.

[21] T. S. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2002.

[22] M. Elliot and T. Stevens, "Dynamic range extension using HARLEQUIN and HAIL," in *MILCOM 2016 IEEE Military Communications Conference*. ieeexplore.ieee.org, Nov. 2016, pp. 835–841.

[23] OptiTrack, "Motion capture systems," http://optitrack.com/, accessed: 2017-7-19.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An open-source robot operating system," in *ICRA workshop on Open Source Software*, vol. 3, 2009, p. 5.

[25] R. Hunjet, T. Stevens, B. Fraser, and P. George, "Survivable communications and autonomous deliver service," in *To appear in MILCOM 2017 IEEE Military Communications Conference*. ieeexplore.ieee.org, Oct. 2017.

[26] Open Source Robotics Foundation, "rosjava - ROS wiki," http://wiki.ros.org/rosjava?distro=kinetic, accessed: 2017-8-23.

[27] D. Gagne, "QGC – QGroundControl – drone control," http://qgroundcontrol.com/, accessed: 2017-7-19.

[28] L. E. Barnes, "A potential field based formation control methodology for robot swarms." ProQuest, Mar. 2008, pp. 27–47.

[29] W. M. Spears and D. F. Spears, *Physicomimetics: Physics-based swarm intelligence*. Springer Science and Business Media, 2012.

[30] M. A. Goodrich, "Potential fields tutorial," *Class Notes*, vol. 157, 2002.

[31] J. C. Barca and K. Acres, "Determining elevation and bearing information of a remote point," (Prov. patent: 2015904219, filed Oct. 15, 2015).

[32] C. Fairchild and T. L. Harman, *ROS Robotics By Example*. Packt Publishing Ltd, 2016.

[33] "Getting started with the Crazyflie 2.0 bitcraze," https://www.bitcraze.io/getting-started-with-the-crazyflie-2-0/.