

# Consensus-Based Data Sharing for Large-Scale Aerial Swarm Coordination in Lossy Communications Environments

Duane T. Davis, Timothy H. Chung, Michael R. Clement, Michael A. Day

Advanced Robotic Systems Engineering Laboratory

Naval Postgraduate School

Monterey, California USA

{dtdavil, thchung, mrclemen, maday@nps.edu}

**Abstract**—Increasing unmanned aerial vehicle (UAV) capabilities and decreasing costs have facilitated growing interest in the development of large, multi-UAV systems, or swarms. The constrained communications environments in which these swarms operate, however, have limited the development of behaviors that require a high degree of deliberative coordination. This work presents two algorithms that use a consensus-algorithm approach to reliably exchange information throughout large swarms as a means of facilitating swarm behavior coordination. Results from experiments conducted in simulation and live-fly exercises are presented and discussed.

## I. INTRODUCTION

In recent years robotic and unmanned systems have been considered to meet the demands of an increasing number of military and civilian applications. Not surprisingly, this trend has only accelerated as robot capabilities have increased and system costs have decreased. The maturation of unmanned systems technology has been particularly evident in the area of unmanned aerial vehicles (UAVs), with low-cost vehicles possessing significant capabilities now available to hobbyists, researchers, corporations, and governments alike.

As individual UAV cost has decreased, interest in cooperative multi-vehicle systems of low-cost UAVs has increased. Much of the current research into these multi-UAV swarms focuses on quad-rotor UAVs, but a number of research efforts are specifically directed at outdoor field experimentation with systems of fixed-wing vehicles. By necessity, many of these research efforts have emphasized platform development [1], [2], [3], use cases [4], [5], and control algorithms for coordinated behaviors [6], [7], [8]. As the numbers of coordinating UAVs has increased, however, additional factors affecting UAV swarms have received increasing attention including human factors associated with control of large swarms [9], [10], logistical aspects of maintaining and deploying these systems [11], and multi-vehicle coordination in the face of communications limits [12], [13]. Inter-vehicle communication can be a particularly vexing problem with regards to coordination for UAV swarms. Issues of latency, synchronization, and reliability can significantly affect the performance of distributed swarm algorithms [13].

Consensus-based approaches have been proposed for a number of multi-UAV coordination problems such as resource and task allocation [14], [15], [16], formation control [17], [18], and determination of agreed-upon coordination variables (e.g., rendezvous time) [19], [20]. These problems share a common thread of requiring eventual

convergence to an agreed-upon solution. Many additional coordination problems, however, do not require formal agreement but can still utilize consensus-algorithm semantics in their solutions. Distributed sorting, for instance, can be applied to swarm behaviors where swarm participants self organize according to criteria derived from individual UAV state characteristics. Rather than relying on an agreed-upon solution, distributed sorting relies on an eventually-consistent swarm-wide understanding of the values being sorted.

As an example, the Naval Postgraduate School (NPS) Advanced Robotic Systems Engineering Laboratory's (AR-SENL) multi-UAV system ensures in-flight deconfliction by vertically separating individual UAVs. Each UAV's position during formation flight is also determined by the commanded altitudes [11]. Actual UAV altitude, however, varies from commanded altitude (due to autopilot limitations, environmental perturbations, etc.) by enough that UAVs occasionally cross altitudes, so current state information is insufficient for reliable sorting. Correct individual UAV positioning within the formation, then, requires swarm-wide exchange of individual commanded altitudes. Unfortunately, one-time or periodic broadcast over lossy communication links will not ensure consistency across the swarm. That is, the ARSENL formation behavior and many other potential behaviors (e.g., search area assignment or rendezvous point based on UAV locations or prioritized landing order based on remaining power) require shared situational awareness that is not provided through unreliable state broadcasts or low cost, on-board sensors typically available to small UAVs. Rather, a reliable and efficient mechanism for behavior-specific information exchange is required to ensure swarm-wide situational awareness sufficient for these behavior-related calculations.

This work proposes a consensus algorithm approach to information sharing to ensure convergence upon a swarm-wide solution for problems relying on distributed UAV-state information. The main contributions of this paper include algorithms that will converge upon swarm-wide agreement on all individual UAV-specific values. These algorithms allow for individual swarm UAVs to reliably determine not only their own roles in a planned swarm behavior, but that of all other swarm UAVs as well. Further, the paper provides an analysis of algorithm performance in real-world and simulated environments and discusses implementation in fail-

stop and fail-silent systems.

An overview of the NPS ARSENL multi-UAV system is provided in Section II, to include a discussion of the system's communications architecture. This system successfully demonstrated the ability to support coordinated UAV swarms of up to 50 UAVs in field exercises [11]. Section III describes the proposed algorithms and the assumptions upon which they rely. The implementation of an *Eventually-Perfect Failure Detector* ( $\diamond P$ ) to facilitate algorithm fault tolerance is also discussed in this section. Analysis of algorithm performance in noise-free simulation, multi-UAV software-in-the-loop (SITL) simulation, and real-world field experiments is provided in Section IV. Finally, conclusions and envisioned avenues of future work are provided in Section V.

## II. THE ADVANCED ROBOTICS SYSTEMS ENGINEERING LABORATORY MULTI-UAV SYSTEM

### A. System Description

The NPS-designed-and-built *Zephyr II* UAV is depicted in Figure 1. With a wingspan of 1.45 meters and a takeoff weight of 2.5 kilograms, the *Zephyr II* has a nominal flight endurance of 50 minutes at a cruise speed of approximately 18 meters per second. To meet cost and performance goals, the airframe relies heavily on open-source and off-the-shelf hobby equipment for flight, avionics, navigation, and communication systems.



Fig. 1. Picture of NPS *Zephyr II* UAV, a low-cost yet capable system leveraging open-source and commercially available components [11].

Swarm behavior deliberative planning and control is conducted on the ODroid companion computer. The ODroid U3 provides computational power similar to that of a modern smartphone and runs the Ubuntu 14.04 Linux operating system. Swarm behaviors are implemented as independent Robot Operating System (ROS) nodes and controlled by a separate swarm-control node. All inter-component communication on the companion computer relies on ROS services and message topics. An autopilot-bridge node provides direction to the Pixhawk Autopilot using the Micro Air Vehicle Link (MAVLink) protocol via a serial link. Network communication between the UAV, ground stations, and other UAVs is handled by a network-bridge node using an ARSENL-developed application-layer protocol. This companion-computer control and communications architecture is depicted in Figure 2.

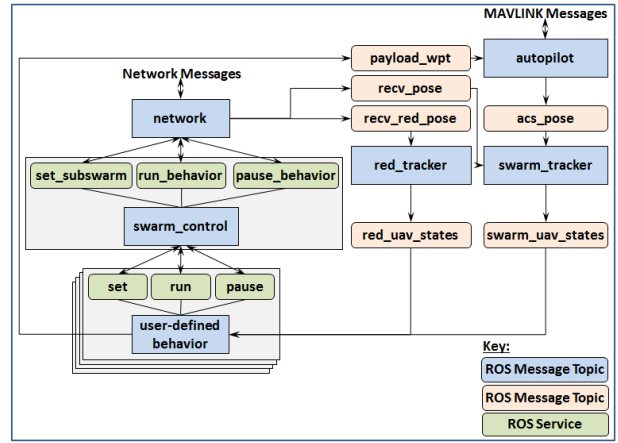


Fig. 2. The ARSENL UAV on-board companion-computer architecture for network communications and swarm behavior activation and deactivation.

The NPS ARSENL team also utilizes an enhanced SITL simulation system for development and testing of single- and multi-UAV algorithms. The SITL environment provides for realistic testing with actual vehicle software in a rigorous, physically-based simulation [21]. The SITL environment also provides the ability to simulate lossy communications environments that are likely to be encountered in real-world scenarios.

### B. Communications Environment

Swarm-wide network communication is implemented over an 802.11n *ad hoc* network using custom application layer messages implemented on top of UDP/IP. Each UAV transmits vehicle-state messages with GPS position, altitude, and linear and angular velocity information at a rate of 10Hz and status messages with subswarm assignment, active swarm behavior, and other information at a rate of 2Hz. The ground control station broadcasts a heartbeat message at a rate of 2Hz. In addition to a set of fixed-format special-purpose messages, customizable swarm-behavior and swarm-data messages are provided to parameterize and initiate swarm behaviors and to conduct inter-vehicle messaging during swarm-behavior execution respectively.

All messages are transmitted as broadcast messages using UDP. As such, all transmitted messages can potentially be received and processed by any swarm UAV, however there is no guarantee of receipt by any UAV. Despite the inherent unreliability of broadcast messaging, it has advantages in the areas of latency and scalability that make it an attractive option for multi-UAV systems. Although it is possible to implement more robust communications primitives such as synchrony, reliable point-to-point, and reliable broadcast [22] using UDP/IP broadcasts, the decision was made to forgo these additions in order to limit additional overhead on the communications infrastructure [11].

Actual network performance for an aerial *ad hoc* network is affected by a number of factors including atmospheric, transmitter strength, transmitter-receiver separation, and relative antenna orientation [13]. Figure 3 provides an exemplar snapshot of UAV-to-UAV packet delivery rates during a 50-

UAV field experiment<sup>1</sup> [11]. At the time depicted, the 50 airborne UAVs were divided into two subswarms of 25 UAVs (subswarm 1 was UAV01-25 and subswarm 2 UAV26-50), each of which was executing ordered swarm behaviors. Same-subswarm UAVs were vertically separated by 15 meters with no lateral deconfliction while inter-subswarm lateral separation varied between approximately 200 and 1000 meters as the subswarms independently executed their ordered behaviors. The figure clearly depicts reduced (but nonzero) delivery rates between subswarms and higher (but not perfect) delivery rates within subswarms.

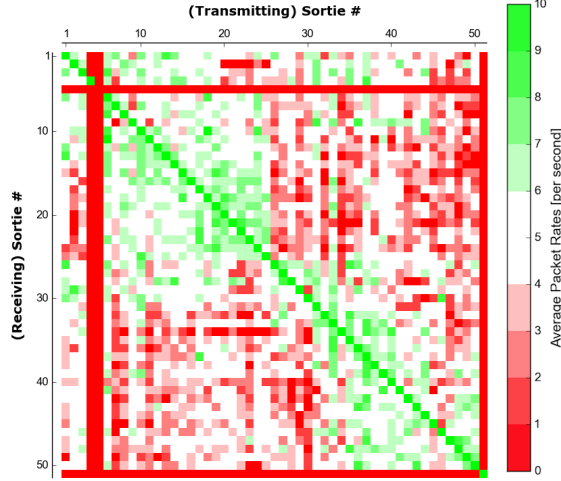


Fig. 3. Aloft network performance as measured by packet rate between aircraft, i.e., packet rate observed at UAV  $i$  (row) received from UAV  $j$  (column) during a 50-UAV flight test. For the period depicted, all 50 UAVs are aloft and operating in two subswarms of 25 UAVs each [11]. Note the communication partitioning between subswarms reflected by the largely green and largely red quadrants.

### III. CONSENSUS-BASED INFORMATION SHARING ALGORITHM DEVELOPMENT

#### A. Networking Model and Underlying Assumptions

Given the observed communications characteristics, it is reasonable to model communication between individual swarm UAVs as fair-loss links. That is, for any particular message transmitted by a UAV,  $j$ , there is a nonzero probability that the message is received by a recipient UAV,  $i$ . This swarm-wide communications model can be expressed in graph form for a swarm of  $n$  UAVs as a weighted adjacency matrix  $\mathcal{A} = [\alpha_{ij}] \in \mathbb{R}^{n \times n}$  where  $\alpha_{ij}$  is the probability that a message transmitted by UAV  $j$  is received by UAV  $i$  [20]. Since network characteristics of an airborne *ad hoc* network vary with individual UAV states, expression of network connectivity as a function of time is appropriate:  $\mathcal{A}(t) = [\alpha_{ij}(t)]$ . Using this time-varying model, individual UAV-to-UAV communications still behave as fair-loss links over time even with  $\alpha_{ij}(t)$  values of zero if it can be assumed that all  $\mathcal{A}(t)$  entries are at least occasionally nonzero. Further, it is

possible, without loss of generality, to discretize the communication model by considering individual communication rounds wherein each UAV may transmit a single message with individual probabilities of receipt for each round  $t$  described by  $\mathcal{A}(t)$ .

This synchronous, discrete communications model is clearly an abstraction of the actual communications characteristics since swarm communications are not formally conducted as fixed-interval rounds. Further, the ARSENL messaging protocol does not enforce message ordering or synchrony [11]. In fact, given the inherent unreliability of UDP/IP broadcast messages, it is unlikely that swarm behavior orders from the ground control station will be received by all swarm UAVs simultaneously. Consequently, the information exchange algorithm cannot implicitly depend on simultaneous swarm-wide initiation or synchronous message exchange. Fortunately, the requirement for the algorithm to be lossy-communications-tolerant also implies tolerance for the lack of synchrony described here. That is, for the purposes of individual UAV execution, rounds occurring before algorithm initiation can be ignored and all other-UAV activity occurring during a single communication cycle can be treated as part of that cycle. Using the time-variant, fair-loss communications model described, the algorithm can be considered active for the entire swarm from the time it is initiated on the first UAV until it is terminated on the last UAV by allowing nonzero adjacency matrix values,  $\alpha_{ij}(t)$ , only if the algorithm is active on UAVs  $i$  and  $j$  at time  $t$ .

Two types of variable-length, inter-vehicle swarm-data messages are utilized by the algorithms described in this work: a request message and a data message. The request message consists of a list of swarm UAV identifiers (2-byte unsigned integers) for which the sending UAV does not have data. The data message consists of a list of UAV-identifier/UAV-data (32-bit floating point) tuples. During each communications round, each UAV has an opportunity to send one request message and one data message and can process all data and request messages received from other UAVs during that round.

Two data sharing algorithms are implemented and tested for this work. Both algorithms require two initialization variables: a *swarm* set, and a *data\_avail* set. The first is a set of all swarm UAV identifiers for which data is required. The second is a set of identifier/data tuples for which the UAV has the required data. At initialization, the *data\_avail* set contains only the executing UAV's own identifier/data tuple. Both algorithms remain active on all swarm UAVs until all UAVs in the swarm have received all required data.

#### B. Lazy Consensus Data Sharing

The first implemented algorithm is referred to as the lazy consensus data sharing algorithm, or simply the lazy algorithm. Depicted in Figure 4, the lazy algorithm operates in four basic steps, the first two of which are only required until the executing UAV has identifier/data tuples from all swarm UAVs.

<sup>1</sup>Note that UAV05 was manually landed and no longer transmitting or receiving state information after being powered down. For further discussion, the reader is referred to [11].

```

1:  $swarm \leftarrow swarm\_uav\_ids$ 
2:  $data\_avail \leftarrow \{own\_data\}$ 
3: repeat
4:   if  $\exists uav \in swarm \wedge uav \notin data\_avail$  then
5:      $new\_data \leftarrow NET\_RCV\_DATA$ 
6:      $data\_avail = data\_avail \cup new\_data$ 
7:      $own\_request \leftarrow swarm \setminus data\_avail$ 
8:      $NET\_SEND\_REQUEST(own\_request)$ 
9:   end if
10:   $requests \leftarrow NET\_RCV\_REQUESTS$ 
11:  if  $own\_data \in requests$  then
12:     $data\_to\_send \leftarrow \{own\_data\}$ 
13:  else
14:     $data\_to\_send \leftarrow \emptyset$ 
15:  end if
16:   $NET\_SEND\_DATA(data\_to\_send)$ 
17: until terminated

```

Fig. 4. The lazy-consensus data sharing algorithm which allows UAVs to provide data in response to requests from other swarm UAVs only when the request is for the responding UAV's own data.

At the beginning of each communications cycle, the lazy algorithm determines whether or not there are swarm UAVs for which the *data\_avail* set does not contain an identifier/data tuple. If this is the case, any data messages received since the last communication cycle are processed and the *data\_avail* set is updated as applicable. It is worth noting that since all messages utilize UDP/IP broadcasts to the entire swarm, individual messages may contain identifier/data tuples that are not required by a particular recipient. After processing all data messages, the algorithm determines which, if any, UAV tuples are still missing from the *data\_avail* set. It then constructs and broadcasts an appropriate request message as required.

Regardless of whether or not all required data are contained in the *data\_avail* set, the lazy algorithm processes all newly-received request messages during every communications cycle. If the executing UAV's identifier is present in any of the received request messages, a data message containing the executing UAV's identifier/data tuple is constructed and transmitted to the swarm.

During the first communication cycle, each UAV's *data\_avail* set contains only its own identifier/data tuple, so a swarm of size  $n$  will require  $n$  request messages, each containing  $n - 1$  identifiers. After the first communications cycle, executing UAVs will broadcast both request messages and data messages as prescribed by the algorithm. Since the lazy algorithm provides a data response only when its own data is requested, all data messages will include exactly one identifier/data tuple. Worst case performance for this algorithm occurs when all UAV requests are received by every swarm UAV, but no data messages are received by any swarm UAV (i.e.,  $\alpha_{ij}(t) = 1$  and  $\alpha_{ij}(t+1) = 0$  for all  $i \neq j$  and a request cycle  $t$  and response cycle  $t + 1$ ). For  $n$  UAVs, this will require  $n$  request messages containing  $n - 1$  UAV identifiers and  $n$  data messages containing a single identifier/data tuple ( $2n$  total messages). The total number of per-round message payload bytes required for the ARSENL implementation of 2-byte UAV identifiers and 8-byte identifier/data tuples for the general case and worst-case

instances is described by Equations 1 and 2 respectively:

$$PB = \sum_{i=1}^n 2r_i + \sum_{i=1}^n 8d_i \quad (1)$$

$$PB_{max}^{lazy} = 2n^2 + 6n \quad (2)$$

where  $n$  is the number of swarm UAVs,  $r_i$  is the number of swarm UAVs for which UAV  $i$  is requesting data and  $d_i$  is the number of identifier/data tuples to be provided in response (1 or 0 for the lazy algorithm). Equation 3 describes the probability that a particular request-response interaction between UAVs  $i$  and  $j$  will be successful (i.e., the probability that UAV  $i$  receives requested UAV  $j$  data as a direct result of the request) given matrices  $\mathcal{A}(t)$  and  $\mathcal{A}(t + 1)$ :

$$P^{lazy}(i \leftarrow j) = \alpha_{ij-jj}(t) = \alpha_{ij}(t) \cdot \alpha_{ji}(t+1) \quad (3)$$

In reality  $\alpha_{ij-jj}(t)$  actually represents the minimum probability that UAV  $i$  will receive the requested UAV  $j$  data. The broadcast communications architecture implies that the actual probability may be somewhat higher since other UAVs may also require UAV  $j$ 's data, and UAV  $j$  will broadcast a data message in response to any relevant request. Nevertheless, since the round-trip probabilities represented by  $\alpha_{ij-jj}(t)$  are themselves semantically equivalent to fair-loss links, the lazy consensus data exchange algorithm will probabilistically converge on a state where all UAV *data\_avail* sets contain identifier/data tuples from all swarm UAVs. Eventual swarm-wide convergence is, however, dependent on a successful directed data exchange between all UAV pairs ( $n^2$  exchanges), so each value in the  $\mathcal{A}(t)$  matrix is effectively a potential single point of failure. Thus, links with low ( $\alpha_{ij-jj}(t)$ ) values for extended periods can delay arrival at a final solution for individual UAVs.

### C. Eager Consensus Data Sharing

The second implemented algorithm, referred to as the eager consensus data sharing algorithm or simply the eager algorithm, is described in Figure 5. The eager algorithm relies upon the same basic steps as the lazy algorithm. In fact, the only difference between the two algorithms is in the response to request messages. Rather than broadcasting a single identifier/data tuple data message only when the executing UAV's data has been requested by another swarm UAV, the eager algorithm will construct and broadcast a data message containing all requested identifier/data tuples that are available in its own *data\_avail* set. This algorithm attempts to leverage overall communications graph connectivity to overcome problematic individual links. The potential improved data flow of the eager algorithm comes at the cost of increased per-round broadcast requirements.

The first request-response cycle of the eager algorithm proceeds in exactly the same manner as the lazy algorithm since each UAV's *data\_avail* set initially contains only its own identifier/data tuple. In later rounds, data messages will grow in size as some individual UAVs' *data\_avail* sets grow faster than others. Per-round message payload

```

1:  $swarm \leftarrow swarm\_uav\_ids$ 
2:  $data\_avail \leftarrow \{own\_data\}$ 
3: repeat
4:   if  $\exists uav \in swarm \wedge uav \notin data\_avail$  then
5:      $new\_data \leftarrow NET\_RCV\_DATA$ 
6:      $data\_avail = data\_avail \cup new\_data$ 
7:      $own\_request \leftarrow swarm \setminus data\_avail$ 
8:      $NET\_SEND\_REQUEST(own\_request)$ 
9:   end if
10:   $requests \leftarrow NET\_RCV\_REQUESTS$ 
11:   $data\_to\_send \leftarrow requests \cap data\_avail$ 
12:   $NET\_SEND\_DATA(data\_to\_send)$ 
13: until terminated

```

Fig. 5. The eager-consensus data sharing algorithm which allows any swarm UAV to provide any requested data that it has available regardless of which UAV the data pertains to or which UAV made the request.

bytes required by the eager algorithm can be obtained from Equation 1 with  $d_i$  values from 0 to  $n - 1$ . Worst-case per-round performance occurs in the pathological case where requests by an individual UAV  $i$  are always received by all swarm UAVs, but no data messages are ever received by UAV  $i$  (i.e.,  $\alpha_{ij} = 1$  and  $\alpha_{ji} = 0$  for  $j \neq i$ ). In this case,  $data\_avail$  sets for all UAVs but one will eventually contain all required tuples. This results in one request message and  $n - 1$  data messages per round with required payload bytes computed with Equation 4:

$$PB_{max}^{eager} = 8n^2 - 14n + 6 \quad (4)$$

Unlike the lazy algorithm, when the  $data\_avail$  set for a UAV,  $i$ , does not contain a tuple for a UAV,  $j$ , any swarm UAV whose  $data\_avail$  set does contain the requested tuple may answer the request. Thus, through inclusion-exclusion, the probability that UAV  $i$  receives one or more responses to a request for UAV  $j$  data is provided by Equation 5:

$$P^{eager}(i \leftarrow j) = \sum_{k=1}^{|J|} (-1)^{k-1} \sum_{D \subseteq J, |D|=k} \prod_{d \in D} \alpha_{id,di} \quad (5)$$

where  $J$  is the set of UAVs with  $data\_avail$  sets containing the requested tuple and  $\alpha_{id,di}$  is computed from Equation 3. As with the lazy algorithm, Equation 5 actually represents a minimum probability. The actual probability will be somewhat higher if multiple UAVs request the identifier/data tuple for UAV  $j$ . Since the  $\mathcal{A}(t)$  matrix describes an eventually-strongly-connected graph over time (an outcome of the fair-loss link model), individual identifier/data tuples for all UAVs,  $i$ , will eventually become available to all other swarm UAVs. Further, because there is no reliance on individual links, swarm-wide convergence can be expected to occur more quickly than with the lazy algorithm.

#### D. UAV Failure Handling

As described thus far, neither the lazy algorithm nor the eager algorithm provide any fault tolerance. Given a set of swarm UAVs, each swarm UAV continues executing either algorithm until the  $data\_avail$  set contains an identifier/data tuple for every swarm UAV. If even a single UAV were to fail after initialization of the  $swarm$  set and before completion of the algorithm, the algorithm may not terminate. More

specifically, if the failed UAV's identifier/data tuple has not been received by every non-failed UAV in the case of the lazy algorithm or at least one non-failed UAV in the case of the eager algorithm, then the algorithm will not terminate. Moreover, even if the failed UAV's tuple is eventually received by all other swarm UAVs, their  $data\_avail$  sets will not accurately represent the current state of the swarm since they will each contain a tuple for the failed UAV. A capability to detect UAV failures, either before algorithm initiation or during execution, would be a clear improvement in the algorithm's robustness.

Reliable failure detection in an asynchronous system such as the communication architecture described here is inherently difficult, primarily because of the difficulty in differentiating between a process that has failed and one that is just slow to respond. The periodic state and status messages broadcast by all UAVs, however, can be utilized to implement an *Eventually-Perfect Failure Detector* ( $\diamond P$ ) meeting certain criteria [22]:

- *Completeness*: Every failed process is eventually permanently suspected of failure by every correct process.
- *Accuracy*: After some period of time, no correct process is suspected of failure by any correct process.

The set of  $\diamond P$  failure detectors has been shown to be sufficient for achieving consensus in a distributed system as long as the majority of processes are correct [22].

The  $\diamond P$  failure detector implemented in support of this work utilizes the periodic state and status messages transmitted by each swarm UAV as described in Subsection II-B. Considering the fair-loss model of the communications architecture, every active UAV can be assumed to receive these messages from every other active UAV at a rate described by the matrix  $\mathcal{A}$ . Upon receipt of a state or status message, the message timestamp is associated with the source UAV's identifier. Periodically, the set of timestamps is tested, and any UAVs for which the time-since-last-update exceeds a *time\_out* value are placed in a *suspect\_crash* set. If a state or status message is received from a *suspect\_crash* set UAV, the UAV is removed from the set, and the *time\_out* value is updated to the observed between-message duration. False suspected crash indications can be expected early in a mission, but over time the *time\_out* values should stabilize to appropriate values for the actual communications environment in which the swarm is operating. This failure detector is equivalent to the time-out based  $\diamond P$  failure detector described in [23] and is implemented within the *swarm\_tracker* node on each UAV in the ARSENL multi-UAV system [11].

Incorporation of the failure detector described here into the lazy and eager data exchange algorithms is fairly straightforward, requiring only removal of *suspect\_crash* set members from the completion test (line 4 of Figures 4 and 5), the request composition (line 7), and the  $data\_avail$  set of the final solution. Since failure detector false positives are possible, and *suspect\_crash* set members may be removed, identifier/data tuples associated with *suspect\_crash* set members



should not be removed from the *data\_avail* set until the final solution is obtained. Implemented as described, this  $\diamond\mathcal{P}$  failure detector reliably accounts for UAV failures that occur before initiation of the algorithm, and failures occurring after commencement of the algorithm are appropriately accounted for if post-failure algorithm execution time exceeds the largest swarm UAV failure detector *time\_out* duration.

Unfortunately, this does leave two potential erroneous termination states. The first is associated with a post-initiation failure of UAV *i*, where at least one UAV, *j*, has received *i*'s identifier/data tuple prior to the failure. In this case, lazy algorithm semantics will prevent premature swarm-wide termination unless all UAVs received the failed UAV's tuple. The termination state of individual UAVs, however, may differ if some include the failed UAV tuple and others do not. The eager algorithm, on the other hand, will forward the failed UAV's tuple among remaining UAVs until it is included in all of their final *data\_avail* sets, resulting in a swarm-wide conclusion that includes the failed UAV.

The second potential erroneous outcome is associated with failure detector false positives. As described in [22] and [23], there is a nonzero probability that a live UAV's state and status messages are not received within an arbitrarily long *time\_out* duration. In the pathological case, both the lazy and eager algorithms can terminate on one or more UAVs without that UAV including the identifier/data tuple for at least one valid swarm UAV.

It should be noted that during testing in simulation, erroneous termination states (all of which were of the failure detector false positive variety) were observed on only three UAVs of a total of 200 tested over the course of 20 experiments with communications loss rates of 90 percent. Further, the real-world communications characteristics observed in [11] indicate that eventual failure detector *time\_out* values are such that false positives among airborne UAVs are unlikely. Specifically, the worst UAV-to-UAV characteristics are observed between airborne and grounded UAVs (i.e., communication is significantly worse between launched UAVs and yet-to-be-launched UAVs than between airborne UAVs). Empirical evidence suggests that time-out based  $\diamond\mathcal{P}$  failure detector false positives are unlikely once all UAVs are airborne.

#### IV. EXPERIMENTAL RESULTS AND ANALYSIS

##### A. Synchronous, Noise-Free Simulation

Both the lazy and eager algorithms were implemented and tested in a MATLAB simulation. Unlike the real-world and SITL simulation tests of the next section, the MATLAB implementation intentionally maintained perfect synchrony among all agents. Thus, with each experiment the tested algorithm was started simultaneously on all executing agents and each communication cycle was allowed to complete for all agents before commencement of the next cycle. This arrangement provided a noise-free environment for assessing algorithm characteristics, performance, and scalability prior to real-world and SITL experiments. The MATLAB simu-

lation was used to test both algorithms with swarm sizes of between 10 and 1000 and communications packet loss rates of 0.25, 0.50, 0.75 and 0.90. Because individual agents were not permitted to fail, the  $\diamond\mathcal{P}$  failure detector was not incorporated into the MATLAB simulation.

Figure 6 (a) and (b) depict average rounds to algorithm convergence for the MATLAB simulation for the eager and lazy algorithms respectively. Not unexpectedly, the eager algorithm reliably converged much more quickly than the lazy algorithm for the same swarm size and packet loss rate. In fact, eager algorithm convergence at the 0.90 packet loss rate met or exceeded lazy algorithm convergence at even the lowest packet loss rates. Intuitively, this is a result of the eager algorithm data propagation scheme; even if a particular identifier/data tuple is successfully passed to a relative few swarm UAVs in early cycles, the number of UAVs from which it can be passed in later cycles is increased. On the other hand, the ability of the lazy algorithm to pass a particular tuple to swarm UAVs that need it remains constant for the duration of the algorithm.

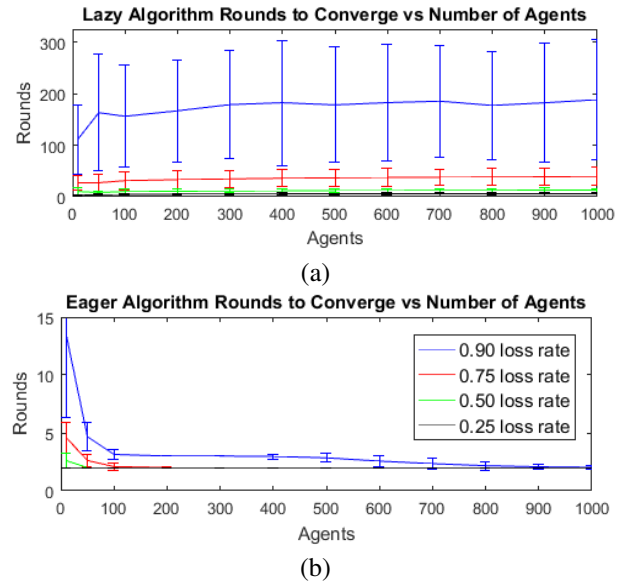


Fig. 6. Number of communications rounds required for eager (a) and lazy (b) algorithm convergence in a MATLAB simulation. Results are depicted for swarm sizes of 10 to 1000 and communications packet loss rates of 0.25, 0.50, 0.75 and 0.90.

It is also evident from Figure 6 that communications packet loss rate only slightly affects eager algorithm convergence and has no discernible effect loss rates above 0.50 (although loss rates above 0.90 may have more significant effects). Lazy algorithm convergence, however, is significantly affected by packet loss rate (showing polynomial growth) and is slightly affected by increasing swarm size.

As described in Section III-C, eager algorithm message payload requirements per round can be higher despite its faster convergence. Figure 7 provides a summary of average message payload per round for 1000-agent simulations of both the lazy and eager algorithms. As expected, the first communication round is identical for both algorithms, as

each agent initially transmits a request message for all other identifier/data tuples and responds with a data message containing only its own tuple. As the algorithm progresses, lazy algorithm message payload requirements decrease consistently over time and approach zero at a rate commensurate with the communications packet loss rate.

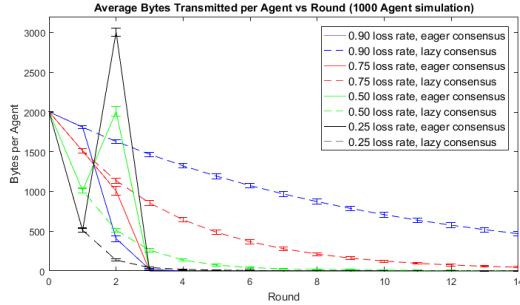


Fig. 7. Average message payload bytes per agent (combined request and data) by round for 1000-agent lazy and eager algorithm simulations with communications packet loss rates of 0.25, 0.50, 0.75 and 0.90.

Eager algorithm experiments with lower communications packet loss rates (i.e., 0.25 and 0.50) indicate a high volume of payload data transmission in the second communication cycle. This can be accounted for by noting that a higher number of tuples are successfully exchanged in the first round, but it is unlikely that any tuple is successfully exchanged with every swarm UAV. More agents will thus be in a position to respond to request messages in round two. Interestingly, at higher packet loss rates (i.e., 0.75 and 0.90), per-agent round-two message payload requirements are actually lower with the eager algorithm than with the lazy algorithm. When compared to lower packet loss rate simulations, fewer agents successfully receive a specific UAV's identifier/data tuple, so the number of agents that are able provide that tuple in response to round-two requests is smaller. As previously observed, this phenomenon does not appear to impact the algorithm's convergence rate.

### B. Software-in-the-Loop and Field Experimentation

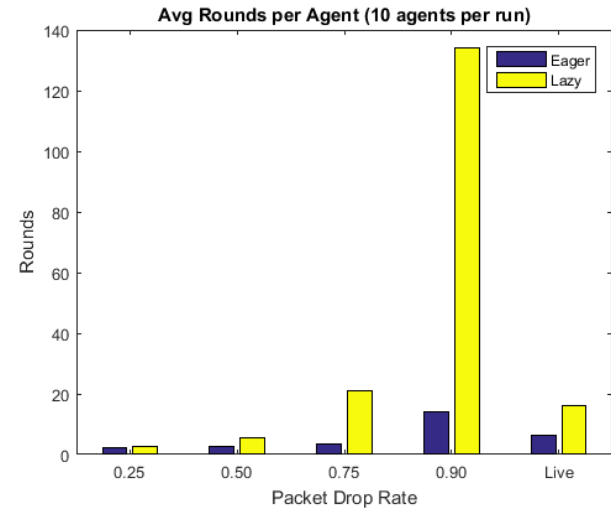
SITL simulation and in live-fly experiments were conducted with a swarm of 10 UAVs<sup>2</sup>. SITL simulations were conducted with communication packet loss rates of 0.25, 0.50, 0.75, and 0.90. Because live and simulated UAV's interact asynchronously, SITL simulation and real-world experiments introduced a number of realistic factors that were not present in synchronous, noise-free tests, such as asynchronous behavior activation and termination and irregular communications cycles in particular.

In addition to testing in isolation, both algorithms were incorporated into more robust swarm behaviors. Specifically, formation flight and a swarm landing behaviors were developed that sort the ordered altitudes of all UAVs to determine

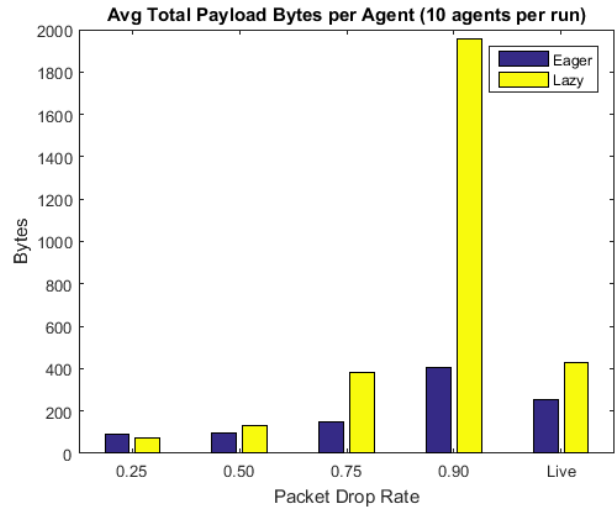
<sup>2</sup>Swarm size for the live-fly experiment was dictated by larger exercise objectives. SITL experiments have been conducted with swarms of up to 50 UAVs, but results are presented for 10-UAV swarms to maintain consistency with the live-fly experiments.

individual UAV formation and landing positions respectively. Both behaviors have been successfully demonstrated in SITL simulations and live-fly experiments [11].

Algorithm convergence and total message payload bytes required are depicted in Figure 8 (a) and (b) respectively. Algorithm convergence aligns with results of 10-UAV noise-free simulations; however, slower eager-algorithm convergence with high packet loss rates, while evident in the synchronous simulation, is more pronounced in SITL experiments. System asynchrony, particularly regarding algorithm initiation, provides a partial explanation (i.e., high loss rates will require more ground-to-air swarm behavior message transmissions before swarm-wide receipt).



(a)



(b)

Fig. 8. Average lazy and eager algorithm convergence (a) and total payload message bytes required (b) for Software-in-the-Loop simulation and real-world experiments for a 10-UAV swarm.

Observed message payload requirements also align with synchronous simulation results. In particular, the total number of message payload bytes required for eager algorithm convergence can exceed that of the lazy algorithm for low data loss rates. At higher communications packet loss

rates, however, lazy algorithm messaging requirements far exceed those of the eager algorithm. Per-round message requirements were similar to those observed in 10-UAV synchronous simulations; however, due to asynchronous SITL and real-world algorithm initiation, lazy and eager algorithm messaging is not identical for the first two rounds as it was in the synchronous tests.

As a final observation, live-fly results seem to correspond to a higher communication packet loss rate than the empirical evidence of Figure 3 implies. Previously discussed asynchrony issues provide a possible explanation; however, rigorous study of this phenomenon remains elusive due to measurement challenges in physical embedded systems. Moreover, the communications environment is but one of many factors affecting swarm scalability [11], so the encouraging synchronous simulation results may or may not be borne out with substantial swarm-size increases.

## V. CONCLUSION

This paper presented a consensus approach to reliable data sharing among UAVs operating in large, cooperative swarms. Inter-vehicle coordination in these systems is hampered by inherently unreliable communications architectures that preclude the exclusive use of periodic broadcasts to exchange required information. Two algorithms were presented: a lazy algorithm that passes information only in direct vehicle-to-vehicle exchange of source vehicle data and an eager algorithm that forwards all available requested information regardless of source. Of these two, the eager algorithm showed faster convergence in both low-loss and high-loss communications environments. Surprisingly, the eager algorithm also required a lower per-round volume of transmitted data in high-loss communications environments. The lazy algorithm required less data transmission per round in low-loss communications environments.

Initial experimentation with these algorithms has been encouraging, but a number of areas of further research are apparent. Algorithms such as these are potentially applicable to a large array of swarm behaviors, and their incorporation into more robust behaviors, possibly in support of or in combination with other distributed algorithms, will be worthwhile. In addition, since reliability was emphasized in their development, improvements to the algorithms themselves can be explored, particularly in efficiency and fault tolerance. Finally, this work will benefit from SITL and real-world testing with larger swarms and more thorough characterization of the communication environments in which multi-UAV swarms are intended to operate.

## REFERENCES

- [1] R. W. Beard, D. Kingston, M. Quigley, D. Snyder, R. Christiansen, W. Johnson, T. McLain, and M. Goodrich, "Autonomous vehicle technologies for small fixed-wing uavs," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 1, pp. 92–108, 2005.
- [2] D. T. Cole, S. Sukkarieh, A. H. Gökdoğan, H. Stone, and R. Hardwick-Jones, "The development of a real-time modular architecture for the control of uav teams," in *Field and Service Robotics*. Springer, 2006, pp. 465–476.
- [3] P. Almeida, R. Bencatel, G. M. Gonçalves, and J. B. Sousa, "Multi-UAV Integration for Coordinated Missions," in *Encontro Científico de Robótica, Guimarães*, Apr. 2006.
- [4] A. Jaimes, S. Kota, and J. Gomez, "An approach to surveillance an area using swarm of fixed wing and quad-rotor unmanned aerial vehicles uav (s)," in *System of Systems Engineering, 2008. SoSE'08. IEEE International Conference on*. IEEE, 2008, pp. 1–6.
- [5] A. Bürkle, F. Segor, and M. Kollmann, "Towards autonomous micro uav swarms," *Journal of intelligent & robotic systems*, vol. 61, no. 1–4, pp. 339–353, 2011.
- [6] J. How, E. King, and Y. Kuwata, "Flight demonstrations of cooperative control for UAV teams," in *In Proc. of AIAA 3rd Unmanned Unlimited Technical Conference, Workshop and Exhibit*, Chicago, IL, Sep. 2004.
- [7] S. Bayraktar, G. E. Fainekos, and G. J. Pappas, "Experimental cooperative control of fixed-wing unmanned aerial vehicles," in *Decision and Control, 2004. CDC. 43rd IEEE Conference on*, vol. 4. IEEE, 2004, pp. 4292–4298.
- [8] C. Kownacki and D. Odziej, "Flocking Algorithm for Fixed-Wing Unmanned Aerial Vehicles," in *Advances in Aerospace Guidance, Navigation and Control SE - 24*, J. Bordeneuve-Guibé, A. Drouin, and C. Roos, Eds. Springer International Publishing, 2015, ch. Flocking A, pp. 415–431.
- [9] H. Wang, S. Y. Chien, M. Lewis, P. Velagapudi, P. Scerri, and K. Sycara, "Human teams for large scale multirobot control," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*. IEEE, 2009, pp. 1269–1274.
- [10] M. Cummings, "Operator Interaction with Centralized Versus Decentralized UAV Architectures," in *Handbook of Unmanned Aerial Vehicles*, K. Valavanis and G. Vachtsevanos, Eds. Springer International Publishing, 2015, ch. UAV Architectures, pp. 977–992.
- [11] T. Chung, M. Clement, M. Day, K. Jones, D. Davis, and M. Jones, "Live-Fly, Large-Scale Field Experimentation for Large Numbers of Fixed-Wing UAVs," in *2016 IEEE International Conference on Robotics and Automation*, Stockholm, Sweden, 2016.
- [12] S. Hauert, S. Leven, M. Varga, F. Ruini, A. Cangelosi, J.-C. Zufferey, and D. Floreano, "Reynolds flocking in reality with fixed-wing robots: Communication range vs. maximum turning rate," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, Sept 2011, pp. 5015–5020.
- [13] I. Bekmezci, O. K. Sahingoz, and Ş. Temel, "Flying ad-hoc networks (fanets): A survey," *Ad Hoc Networks*, vol. 11, no. 3, pp. 1254–1270, 2013.
- [14] L. Brunet, H.-L. Choi, and J. P. How, "Consensus-based auction approaches for decentralized task assignment," in *AIAA Guidance, Navigation, and Control Conference, Honolulu, Hawaii*, 2008.
- [15] S. Gruber, R. Streeter, and G. York, "Flexible multi-agent algorithm for distributed decision making," in *Unmanned Aircraft Systems (ICUAS), 2015 International Conference on*. IEEE, 2015, pp. 9–17.
- [16] S. S. Ponda, L. B. Johnson, A. Geramifard, and J. P. How, "Cooperative mission planning for multi-uav teams," in *Handbook of Unmanned Aerial Vehicles*. Springer, 2015, pp. 1447–1490.
- [17] W. Ren, R. W. Beard, and E. M. Atkins, "A survey of consensus problems in multi-agent coordination," in *American Control Conference, 2005. Proceedings of the 2005*. IEEE, 2005, pp. 1859–1864.
- [18] Y. Kuriki and T. Namerikawa, "Experimental validation of cooperative formation control with collision avoidance for a multi-uav system," in *Automation, Robotics and Applications (ICARA), 2015 6th International Conference on*. IEEE, 2015, pp. 531–536.
- [19] I. H. Makhdoom and Q. Shi-Yin, "Simultaneous arrival of multiple uavs under imperfect communication," *Aircraft Engineering and Aerospace Technology*, vol. 84, no. 1, pp. 37–50, 2012.
- [20] X. Wei, D. Fengyang, Z. Qingjie, Z. Bing, and S. Hongchang, "A new fast consensus algorithm applied in rendezvous of multi-uav," in *Control and Decision Conference (CCDC), 2015 27th Chinese*. IEEE, 2015, pp. 55–60.
- [21] M. A. Day, M. R. Clement, J. D. Russo, D. Davis, and T. H. Chung, "Multi-UAV Software Systems and Simulation Architecture," in *2015 International Conference on Unmanned Aerial Systems*. Denver, CO: IEEE, 2015, pp. 426–435.
- [22] T. D. Chandra and S. Toueg, "Unreliable failure detectors for reliable distributed systems," *Journal of the ACM (JACM)*, vol. 43, no. 2, pp. 225–267, 1996.
- [23] D. Dolev, C. Dwork, and L. Stockmeyer, "On the minimal synchronism needed for distributed consensus," *Journal of the ACM (JACM)*, vol. 34, no. 1, pp. 77–97, 1987.