Flight Test Results of a Distributed Merging Algorithm for Autonomous UAS Operations

Andrew Peters
National Institute of Aerospace
Hampton, VA, USA
andrew.peters@nianet.org

Swee Balachandran
National Institute of Aerospace
Hampton, VA, USA
swee.balachandran@nianet.org

Brendan Duffy
National Institute of Aerospace
Hampton, VA, USA
brendan.dufy@nianet.org

Kyle Smalling
National Institute of Aerospace
Hampton, VA, USA
kyle.smalling@nianet.org

María Consiglio
NASA Langley Research Center
Hampton, VA, USA
maria.c.consiglio@nasa.gov

César Muñoz
NASA Langley Research Center
Hampton, VA, USA
cesar.a.munoz@nasa.gov

Abstract - **This paper reports the flight test results of an on-demand, distributed, consensus based merging algorithm for autonomous multi-agent coordination used to regulate flow of air traffic through a common intersection or merge fix in a given airspace. Distributed merging is enabled by vehicle-to-vehicle (V2V) communication technology, a distributed consensus algorithm and a scheduling algorithm to coordinate the arrival times of the vehicles approaching a merge fix. The proposed algorithm is integrated into the ICAROUS (Independent Configurable Architecture for Reliable Operations of Unmanned Systems) software suite and was used to demonstrate the merging capability in a flight test campaign. The objectives of these flight tests were to validate the distributed consensus-based merging algorithm and to evaluate the requirements and associated challenges in ensuring the successful application of the algorithm in a real-world setting. Details of the flight test setup, hardware used, impediments to the achievement of the outlined objectives, flight test results and lessons learned are documented in this paper.**

Keywords—merge, ICAROUS, distributed consensus, RAFT, UAS, flight testing

I. INTRODUCTION

The growing potential of large-scale high-density operations of small Unmanned Aircraft Systems (sUAS) to be conducted at locations like warehouse distribution centers for package delivery has driven the need for technologies that safely and efficiently deconflict the flight paths of incoming and outgoing aircraft. While centralized scheduling and planning of operations will ensure strategic deconfliction of flight plans, a distributed, autonomous, onboard vehicle-to-vehicle coordination capability for merging and spacing will provide tactical deconfliction of converging flights.

A series of test flights was conducted at NASA Langley Research Center (NASA LaRC) under the auspices of NASA's UAS Traffic Management (UTM) project to demonstrate the initial capabilities of a merging application developed to enable on-demand deconfliction of intersecting traffic flows at merge fixes. The merging application was integrated into the ICAROUS software suite and Vehicle to Vehicle (V2V) communication technology enabled vehicles to exchange the information required to achieve arrival time deconfliction. A distributed consensus and a scheduling algorithm work in tandem to enable each vehicle to compute coordinated arrival times that are spaced by a predefined separation time interval. These initial flight tests demonstrate three things; a method for deconflicting multiple flight plans, the ability to adjust aircraft velocities to meet that schedule, and finally returning an aircraft to normal operation after proceeding through a merge fix while maintaining a safe horizontal separation distance.

Effective implementation of the merging application required the evaluation of multiple new technologies and operational procedures. The effectiveness of V2V technologies based on 915 MHz Ethernet radio and cellular data links was evaluated. Also, technologies that facilitate command and control of multiple teams operating individual aircraft working together to enable the autonomous maneuvers were evaluated. Traditional sUAS ground stations were used for individual aircraft monitoring. A web based multi-aircraft ground station called WebGS, developed at NASA Langley Research Center, was used to integrate incoming data streams from multiple UASs and provide an enhanced situational awareness of the merging flight test operations.

This paper is organized as follows. Section II provides a background of the various underlying technologies. A brief introduction to ICAROUS and the underlying algorithms used for merging is discussed. A survey of available literature reveals the absence of any work related to flight testing of a distributed merging algorithm for sUAS applications. Section III documents the flight test setup, the hardware and software configurations used in these tests. Section IV documents the outcomes of these tests and analyzes flight tests results. Section

V provides a discussion on lessons learned and improvements necessary to ensure successful autonomous merging. Section VI provides conclusions and outlines future research directions.

II. BACKGROUND

A. ICAROUS

ICAROUS (Independent Configurable Architecture for Reliable Operations of Unmanned Systems) [1,2] is a distributed software architecture designed to provide UAS with decision making capabilities to operate autonomously. ICAROUS is publicly available under NASA's Open Source Agreement at http://github.com/nasa/icarous. The core functionalities include maintaining safe separation with other intruders in the airspace, conforming to geospatial airspace constraints (keep-in/keep-out geofences), avoiding obstacles and performing route planning when alternate flight routes are essential to accomplish mission goals (path planning).

ICAROUS is implemented using the NASA core Flight Systems (cFS) middleware. Each ICAROUS functionality is developed as an independent cFS application. cFS applications can interact with each other by publishing or subscribing to messages on the cFS software bus. The distributed nature of this architecture enables the easy integration of new functionality as modular cFS applications. The cFS software bus can be extended over a network enabling multiple instances of cFS nodes to communicate with each other. This specific feature enables multiple vehicles running ICAROUS to interact with each other over a network connection. Fig. 1 provides a schematic representation of the underlying framework that enables communication between multiple vehicles.

B. Distributed Merging Algorithm

A detailed presentation of the algorithm used to enable distributed merging can be found in [3]. The distributed merging algorithm consists of three main components: (1) A scheduling function which takes as input the earliest and latest feasible arrival time for each vehicle approaching a merge fix, (2) A mechanism to exchange the earliest and latest arrival times among vehicles so that there is consensus on the inputs used by for scheduling and (3) A set of operating procedures that dictate when each vehicle approaching the merge fix is required to exchange data, start scheduling and execute trajectory changes required to reach the merge fix at the scheduled time.

The volume of airspace around the intersection or merge fix through which vehicles are required to coordinate their arrival is abstracted into three concentric zones. The outermost zone or coordination zone, a middle zone or schedule zone and an inner most zone or entry zone. The first vehicle approaching the intersection/merge fix enters the outermost zone (coordination zone) and establishes a network to enable information exchange. Subsequent vehicles join the established network and exchange information about their possible arrival times at the merge fix. Data is exchanged on the network by means of a leader-follower heartbeat mechanism (similar to the RAFT protocol [4]). Nodes (vehicles) in the network vote to elect a leader and the leader ensures the synchronization of data exchanged between nodes. When the vehicles enter the middle zone (schedule zone), they use all the received arrival times to compute an optimal schedule of arrival for all the vehicles participating in that intersection/merge fix. All vehicles compute the schedule using the same input information and hence the scheduled arrival times for all the nodes in the network are guaranteed to be the same across each node. Vehicles start making trajectory changes (speed adjustments) upon entering the innermost (entry zone) to reach the merge fix at the scheduled time.

C. WebGS

WebGS is a web-based ground station designed for multi-aircraft operations. It provides situational awareness and the ability to monitor multiple aircraft simultaneously. WebGS is publicly available under NASA's Open Source Agreement at http://github.com/nasa/webgs. WebGS uses MAVLink [5] messaging protocol to communicate with each aircraft. It uses a combination of multi-processing and asynchronous functionality to simultaneously receive and process multiple telemetry streams. WebGS was designed to be used for planning, automation, simulation, and visualization of UAS flight operations. Fig. 2 is a screen shot from WebGS while monitoring three aircraft in a merging operation. It shows the three merging zones and specific aircraft information for the selected vehicle. In these flight tests, use of WebGS was limited to visualization of the airspace, processing telemetry streams from each aircraft and keeping track of status updates from each aircraft.

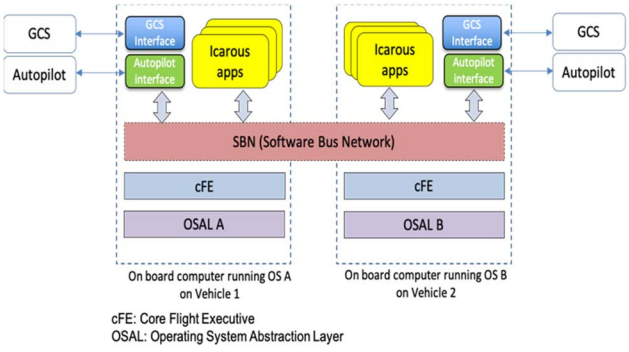

Fig. 1. Multi-vehicle interaction through cFS and SBN

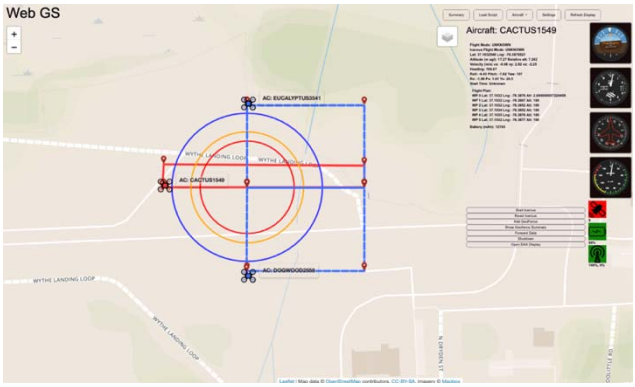

Fig. 2. WebGS display showing three aircraft, flight plans, and merge zones.

III. FLIGHT TEST METHODS

A. Aircraft Configuration

These flight tests consisted of two small UAS (named CERF and ISAAC) and one simulated vehicle (SIM) using ArduCopter Software-In-The-Loop (SITL) simulator. Each small UAS consisted of a DJI S1000 Octocopter [6] frame, Pixhawk autopilot running ArduCopter [7], a companion computer - Intel NUC [8], a RFD-900 radio outputting a 900 MHz telemetry link (Pixhawk output) to Mission Planner ground stations, and a 2.4 GHz command and control link for safety pilots monitoring the operations. A third intel NUC was used as a research ground station and was also used to run the simulated UAS. The simulated UAS consisted of the ICAROUS software stack interacting with the ArduCopter Software In the Loop (SITL) simulator.

All UAS (including the simulated aircraft) had a Verizon Jetpack MiFi 4G LTE cellular hotspot [9] onboard for V2V communications. The Jetpacks were tethered through USB connection to the companion computer with Wi-Fi disabled. Each Jetpack device was configured with a static IP address to enable the vehicles to find each other over the cellular network. Research telemetry (ICAROUS output) from each aircraft, received via the V2V link, was monitored using MAVProxy on the research ground station. Each instance of MAVProxy forwarded the telemetry data to WebGS, which presented the combined vehicle data on a single display as illustrated in Fig. 3.

B. Flight test scenarios

Testing was conducted over two days of operations at the CERTAIN range at NASA Langley Research Center and broken down into eight scenarios. The scenarios were designed to test a combination of varying entry speeds into a merge fix airspace, ranging from 3 to 7 m/s. Table 1 shows the starting point and initial velocity for each aircraft. The start points in Table 1 correspond to the locations shown in Fig. 4. All flights were conducted with 50 meters of vertical separation between the real vehicles. Specifically, CERF and the sim vehicle were flown at an altitude of 50 m above ground level (AGL) while ISAAC was flown at an altitude of 100 m (AGL). The start points of each vehicle were chosen such that failure to autonomously coordinate their arrival at the merge fix would result in a mid-air collision, if not for this vertical separation, which was included for safety. For the purpose of these flight tests, vertical separation between vehicles were ignored when computing conflicts at the merge fix.

The merging scenarios consist of three aircraft heading towards a shared waypoint (Fig. 4, Point D) considered to be the merge fix. To avoid collision, the aircraft coordinate a schedule and make an initial adjustment to their ground speeds to meet the scheduled arrival times at this merge fix. Each vehicle was given a box shaped flight plan that circled back after the merge was completed which allowed multiple merges to be attempted per flight. After takeoff, each vehicle was flown to the start point manually by their respective safety pilots. All vehicles were switched to autopilot control simultaneously and flew towards the merge fix autonomously. To ensure sufficient spacing between each vehicle after exiting the merge fix, all vehicles flew at a constant speed (chosen according to a user defined parameter) upon exiting the merge fix. After each merge attempt the aircraft were reset and the scenario was repeated. Up to four merge attempts were possible in a single flight before a battery change was required.

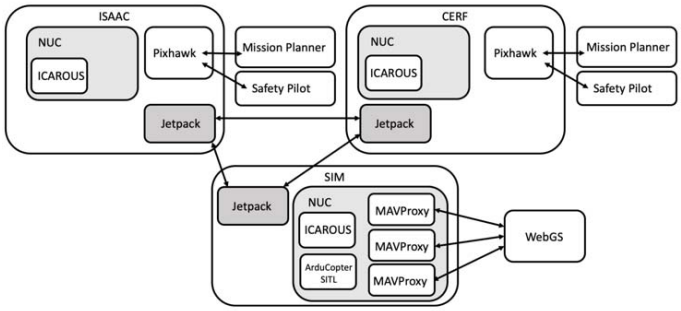

Fig. 3. Flight test configuration.

TABLE I. STARTING POSITIONS AND INITIAL VELOCITIES FOR EACH SCENARIO. SEE FIG. 4 FOR START POINT LOCATIONS.

Flight Plans						
	ISAAC		CERF		SIM	
Scenario	*Start Point*	*Start Speed (m/s)*	*Start Point*	*Start Speed (m/s)*	*Start Point*	*Start Speed (m/s)*
1	B	3	C	3	A	3
2	B	3	C	5	A	5
3	B	3	C	5	A	3
4	B	5	C	5	A	3
5	B2	7	C	5	A	5
6	B2	7	C1	3	A	5
7	B1	3	C	5	A	5
8	B2	7	C2	7	A	5

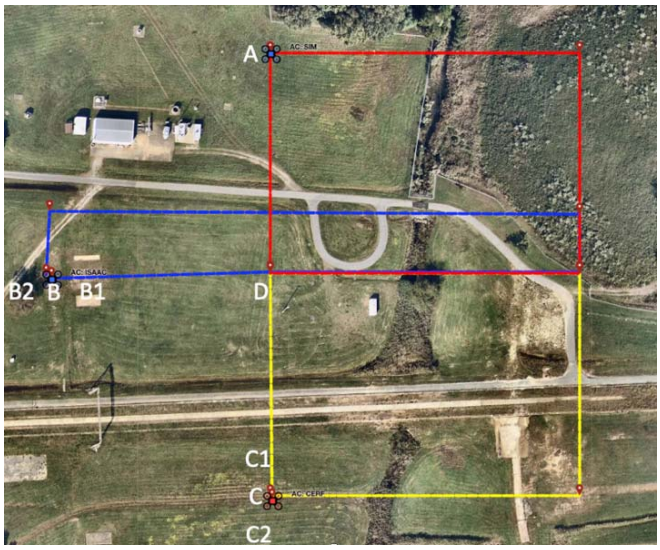

Fig. 4. Flight Plans used at NASA's CERTAIN range.

IV. FLIGHT TEST RESULTS

For the purposes of these preliminary flight tests, a merging encounter was considered successful when all three vehicles communicated with each other, coordinated and adjusted speeds as needed, maintained proper time-based separation, and returned to autopilot control upon reaching the merge fix. A summary of the overall results is first presented followed by an analysis of the successful and failed merging encounters.

A. Overall Results

Over a two-day period, 12 flights were conducted covering all 8 scenarios and repeating failed attempts at scenarios 2, 5 and 6. In the course of the 12 flights, 38 merging encounters were attempted. Of these, 21 were considered successful as per the definition of success outlined above, a success rate of 55%. Day one consisted of five flights. There were successful merges on nine of fifteen attempts, a 60% success rate. On day two, seven flights were conducted and 12 of 23 merges, or 52%, succeeded. On day one scenarios 1, 2, 3, 4, and 7 were flown and on day two scenarios 2, 5, 6, and 8 were flown, and 2, 5, and 6 were repeated.

Two sets of parameter configurations were used across these flights. In configuration A, the minimum and maximum limits on the possible merge speeds used by the UAS were set to 0.5 m/s and 7.0 m/s respectively. The radius of the coordination, schedule and entry zones were set to 80, 60 and 50 meters respectively. The mission speed parameter, which determined the vehicle's speed upon exiting the merge fix was fixed at 3.0 m/s. In configuration B, the minimum and maximum limits on the possible merge speeds used by the UAS were set to 0.5 m/s and 8.0 m/s respectively. The zones were expanded to 100, 75, and 50 meters respectively. The mission speed parameter was adjusted to match the initial speed of the aircraft for each flight (as defined in the flight plan). The relationship between speeds, parameter configuration, and test outcomes can be seen in Table 2.

B. Analysis of Failed Merges

The issues encountered on the failed merges can be broken down into three categories; communication related, inadequate parameter settings and incorrect schedule computation. There were three failures related to communication issues, eight failures due to invalid parameter settings, and six failures related to scheduling.

The first of the three communication issues was related to the loss of telemetry link between the ground station and one of the aircraft resulting in an automatic return to launch (RTL). The RTL behavior is a built-in safety feature that enables vehicles to return to the initial starting location when the autopilot observes a loss in command and control or telemetry link. The second communication issue was related to an intermittent loss of onboard power to the companion computer, resulting in a reboot during the merging encounter thus disrupting the vehicle to vehicle communications. The third failure occurred due to the incorrect positioning of the safety cutoff switch on one of the vehicles R/C transmitter prior to the flight. The safety cutoff switch is a relay designed to sever all communications between the onboard companion computer and autopilot system that is operated by the pilot. These failures have no real bearing on the merging application.

Some of the failed merges were attributed to the minimum merge speed parameter being set too low for the real aircraft. As the initial flight speeds increased on scenarios five and six, some of the aircraft were being scheduled lower merging speeds to allow sufficient separation between the arrival times of vehicles. The lower merging speeds illuminated a discrepancy in the criteria used by the merging algorithm vs the autopilot to determine if a given waypoint was reached. The result was the aircraft stopping just before reaching the merge fix and not continuing on the flight plan. The minimum speed parameter was adjusted over the following two flights to find a correct setting that would avoid this behavior and it took four successive merge attempts to find a suitable minimum speed setting.

Another parameter issue encountered on some of the failed merges was setting the scheduling zone size too small for the aircraft velocity. This didn't allow enough time for the aircraft to compute, communicate, and agree upon a schedule before the aircraft entered the entry zone. On day one the scheduling zone was 10 m across. At 5 m/s each vehicle was in this zone for 2 s.

TABLE II. VELOCITY AND PARAMETER CHANGES PER FLIGHT.

Flight	CERF (m/s)	ISAAC (m/s)	SIM (m/s)	Attempts	Success	Parameter configuration	Notes
1	3	3	3	3	3	Config A	All merges successful
2	5	5	5	2	0	Config A	Inadequate parameter configuration
3	5	3	3	4	3	Config A	RTL triggered on ISAAC
4	3	5	3	3	2	Config A	Incorrect schedule computation
5	5	3	5	3	1	Config A	Incorrect schedule computation
6	5	5	5	2	1	Config B	NUC Crash
7	5	7	5	4	1	Config B	Research Cutoff/Control issues
8	3	7	5	3	0	Config B	Control issues
9	3	7	5	3	2	Config B	Control issues
10	7	7	5	4	4	Config B	All merges successful
11	5	7	5	4	1	Config B	Incorrect schedule computation
12	5	5	5	3	3	Config B	All merges successful

Any delay in start time or latency in the issuance of commands would decrease the time all vehicles were in the scheduling zone together, thus reducing the time allowed to compute a schedule. Changing to parameter configuration B on day two alleviated some these issues but may have been a factor at higher speeds and could have contributed to the scheduling failures.

Finally, there were six merges that failed to schedule suitable arrival times at the merge fix; Three of these merge failures were attributed to lack of heartbeat acknowledgements between the leader and follower nodes in the network, which in turn could signify a potential vehicle to vehicle communication issue. The root causes of other failures were unable to be identified due to a lack of sufficient data. These errors could have been caused by disruptions in vehicle to vehicle communication or by the lack of sufficient time to compute a schedule when flying at higher speeds. For example, with a scheduling zone that is 10 meters wide a UAS traveling at 7 m/s would have only 1.4 seconds to compute, communicate, and agree upon a schedule.

C. *Analysis of Successful Merges*

This analysis looks at the different components of the merge and the overall performance. Each merge consists of a schedule and a commanded velocity, while requiring the vehicles to maintain a safe separation while merging. This leads to a few questions. Did the aircraft execute the commanded speed change? Did the aircraft arrive at the merge point at the scheduled time and in the proper order? Did the aircraft maintain proper time separation? And did the aircraft maintain a safe minimum horizontal distance? Due to data limitations, analysis of successful merges is limited to day two flights which includes 12 merges from 6 flights.

To understand how well the commands were executed comparing the commanded and actual horizontal velocity gives us a better perspective of the hardware capabilities. The average horizontal velocity was calculated from the time the velocity change command was given to the time control was given back to the autopilot. This was subtracted from the commanded velocity. Fig. 5 shows the changes in velocity for an individual merge. The aircraft was switched into auto from a hover. It then accelerated to its assigned velocity, for this flight it was 7 m/s. At ~840 s it entered the entry zone and was commanded to change to a velocity of 1.21 m/s. At ~882 the aircraft reached the merge fix and returned to its initial velocity. The mean velocity was calculated from the time the command was given t_c to the arrival at the merge fix t_n. The average for each aircraft was calculated, as reported in Table 3. The simulated vehicle was better at matching and holding a commanded velocity. CERF was commanded the highest velocities on most of the merges, followed by ISAAC, then the SIM.

Comparing actual versus scheduled arrival times shows all aircraft arrived in their scheduled order. The first two vehicles were able to arrive an average of 0.25 seconds after the scheduled time, as shown in Table 4. The third vehicle, SIM on all of the successful flights, consistently arrived early to the merge point. This is inconsistent with the SIM's ability to fly a desired velocity better than the real vehicles as shown in Table 3. This may be accounted for by the extremely slow velocities SIM was commanded to fly, as low as 0.8 m/s, the distance

$$d = v_{commanded} - \mu_{v_{\{t_c,\ldots,t_n\}}} \quad (1)$$

TABLE III. MEAN DIFFERENCE BETWEEN COMMANDED MERGE SPEED AND ACTUAL MEAN HORIZONTAL VELOCITY DURING THE MERGE AND STANDARD DEVIATION FOR EACH AIRCRAFT.

Aircraft	μ_d (m/s)	σ_d
ISAAC	0.10	0.135
CERF	0.11	0.240
SIM	0.03	0.050

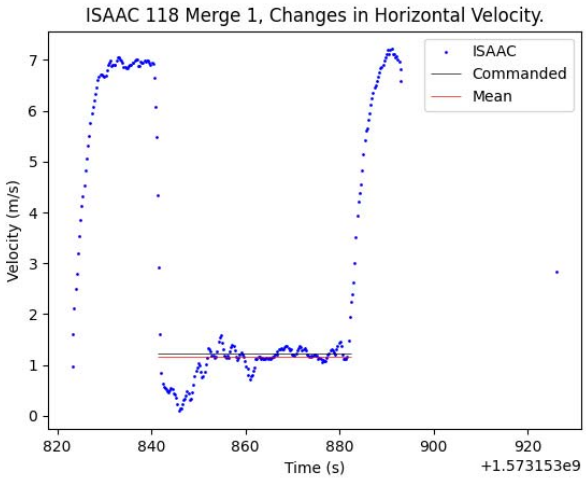

Fig. 5. Changes in Horizontal Velocity.

expected to cover, and since the algorithm does not adjust velocities after the initial command to ensure proper arrival times a small error in commanded velocity would be compounded over time. Another factor may be the time between issuing the command and execution of that command would have a disproportionate effect on slower velocities. This delay in execution may account for part of the discrepancy between scheduled and actual arrival times.

The time difference of arrival (TDOA) between the first and subsequent vehicles shows the same result as actual versus scheduled arrival times. The second vehicle was expected to arrive 20 seconds after the first and the third 40 seconds after the first. Fig. 6 shows the distance from each aircraft to the merge fix for an individual merge. The change in slope at ~780 s indicates a change in commanded velocity and the change from a negative to positive slope indicates the arrival time at the merge fix then flying towards the next waypoint. Table 5 shows on average the second vehicle arrived 20.43 s after the first aircraft and the third arrived 33.82 s after the first.

The horizontal distance between each aircraft was calculated at each time an updated position message was received by the ground station. This distance was based on the last known position of the other aircraft, based on the position messages passed between vehicles and relayed to the ground station at a rate of 1 Hz. Fig. 7 shows the horizontal distance between each aircraft reaching a minimum as they approach the merge fix. This minimum was calculated for each successful merge and

averaged based on arrival order and shown in Table 6. The horizontal distance between the first and second aircraft to arrive averaged 27.73 m, while the average for the second to third was 13.80 m. These values deviated considerably due to variations in commanded velocity and are in a large part a product of the flight plan geometries.

$$T = t_{scheduled} - t_{actual} \quad (2)$$

TABLE IV. MEAN DIFFERENCE BETWEEN SCHEDULED AND ACTUAL ARRIVAL TIME.

Arrival Order	μ_T (s)	σ_T	Min(T) (s)	Max(T) (s)
1	0.25	0.97	-1	+2
2	0.26	1.70	-4	+3
3	-6.33	2.53	-10	-1

$$TDOA = t_{ac_1\,Arrival} - t_{ac_n\,Arrival} \quad (3)$$

TABLE V. TABLE 5. MEAN TIME DIFFERENCE OF ARRIVAL BASED ON ARRIVAL ORDER.

Aircraft	μ_{TDOA} (s)	σ_{TDOA}
2	20.43	2.45
3	33.82	2.66

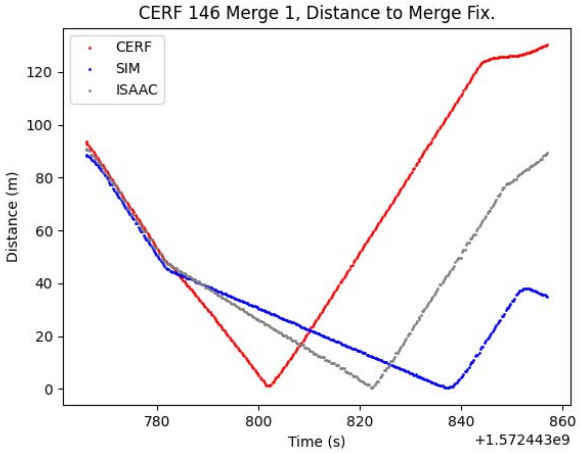

Fig. 6. Distance to Merge fix.

$$d = \min(P_{ac_n} - P_{ac_{n+1}}) \quad (4)$$

TABLE VI. AVERAGE MINIMUM HORIZONTAL DISTANCE ON SUCCESSFUL MERGES.

Aircraft	μ_d (m)	σ_d
1st to 2nd	27.73	3.94
2nd to 3rd	13.80	6.85

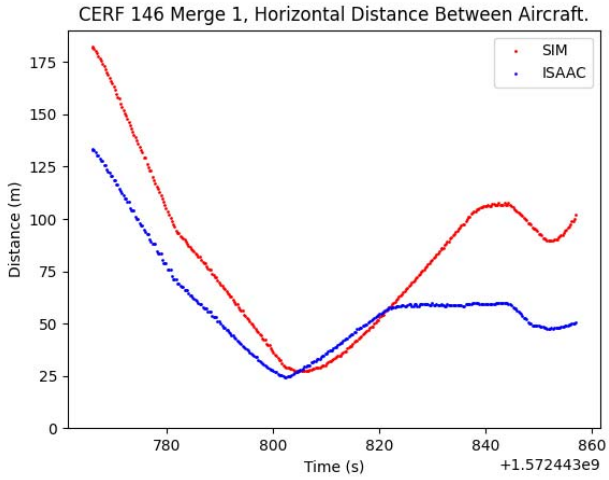

Fig. 7. Horizontal Distance Between Aircraft.

D. *Evaluation of Vehicle to Vehicle Communication Technologies*

Multiple options were considered to provide vehicle-to-vehicle communication to enable information exchange between vehicles. Three different radios were tested; AvaLAN AW900MTR, RFD900x setup in mesh mode, and finally the Verizon 4G MiFi. Initial tests used AvaLAN AW900MTR network radios [10]. These radios enable the transmission of network packets over 900 MHz thus allowing ICAROUS-equipped vehicles to share coordination messages over the cFS software bus network (SBN).

In initial checkout flights, the AvaLAN radio link was intermittent and unreliable. ICAROUS uses User Datagram Protocol (UDP) for it's network communications; this protocol broadcasts messages to all devices participating in the merge and there is no confirmation that the messages have been received by other nodes. Any loss of messages causes the SBN to disconnect and required restarting the research software. This inconvenience made flight testing merging operations difficult, but eventually many merges were executed between a real vehicle and a simulated vehicle using the AvaLAN connection. Adding a second real vehicle (expanding the network to three radios) proved to be too many, and dropouts were so frequent that no successful merging operations were flown in this configuration.

The second radio tested was an RFD900 with the Asynchronous Mesh firmware. The same RFD900 radios with a different firmware (SiK) were being used successfully for the autopilot telemetry stream. When these radios were added for use as the research radio though, none of the radios (including the autopilot telemetry stream) worked reliably. It was found that the RFD900's were only compatible with the SBN serial module which makes assumptions for a point-to-point topology. This allowed communication between two vehicles but no more. The AvaLAN and MiFi radios in contrast were compatible with the SBN UDP module which allowed peer-to-peer networking.

A total of five radios were used during the dual vehicle flights and eight were used during the three vehicle flights. When using the AvaLan and RFD900 research radios, all radios were in the 900MHz band and interference is a likely factor in the poor link quality. Some improvement was seen after using AvaLAN's built in spectrum analyzer and configuring the AvaLAN radios to use the channels with the least activity. However, the improvement was not sufficient to enable three-vehicle operations. Further work could be done to remove interference and optimize radio performance. However, in order to continue testing, the radios were replaced with Verizon 4G MiFi hotspots, enabling vehicle-to-vehicle communication over Verizon's cellular network. This connection proved to be much more reliable and was used for all merging tests.

V. DISCUSSION

The preliminary merging flight tests described in this paper were a proof of concept intended to demonstrate the feasibility of a decentralized, consensus based coordination strategy to ensure vehicles can deconflict arrival times at a merging fix. Testing the merging algorithm with UAS on actual hardware revealed a number of challenges that were not encountered during simulation studies. Biggest of these challenges was the lack of options to support robust communication among multiple vehicles. Furthermore, this also emphasized the importance of a robust communication link between vehicles to enable the timely exchange of information required to coordinate a successful merge.

Additional simulation and flight test studies are essential to understand the scalability limitations of the merging algorithm with respect to the number of vehicles approaching the merge fix, communication throughput and latency. Knowledge of these limitations can help define operational constraints in terms of total number of allowed aircraft in a given time for the given airspace, and minimum and maximum airspeed constraints to ensure a successful merge. Although these flight tests relied on cellular V2V technology for vehicle coordination, suitability of the communication technology must be evaluated with respect to operating environment (e.g. urban canyon vs open airspace).

In multi-UAS operations, the ability of a single operator to understand the context is impossible with currently available off the shelf tools such as Mission Planner and MAVProxy. These tools show an ownship centric view of the operational environment and are not intended to handle more complex scenarios like those conducted in the flight tests described in this paper. During these flights, a researcher monitoring the operations was required to launch one instance of MAVProxy per UAS. Consequently, the operator had to toggle between three instances of MAVProxy to interact with each UAS and monitor the merging operations. Monitoring large swarms of UAS can become cumbersome without a suitable coherent interface to interact with each individual UAS.

VI. CONCLUSION

This paper presented the results of a set of flight test experiments to evaluate an autonomous, decentralized, consensus based merging algorithm. Preliminary results establish that it is possible for sUAS vehicles, with limited computing power, to successfully coordinate their arrival at a common merge fix given a range of starting velocities and starting distances to the merge fix. Although several merge encounters in this flight test campaign were not considered successful as per the definition of success identified in this paper, the majority of the failed encounters were attributed to factors not directly related to the merging algorithm. Furthermore, analysis of the flight data from the failed merge encounters helped resolve potential drawbacks of the merging algorithm. The presence of a robust communication link is essential to achieve coordination among vehicles. Future work aims to study the suitability of the proposed algorithm to support large volumes of UAS and Urban Air Mobility (UAM) type operations.

ACKNOWLEDGEMENTS

The authors would like to specifically thank Kaveh Darafsheh (NASA Langley), Nick Rymer (National Institute of Aerospace), Chris Manderino (University of Pittsburgh) for their contributions to the success of this flight campaign. The authors would also like to thank the NASA LaRC System Wide Safety project team for partnering with us for these flight operations.

REFERENCES

[1] María Consiglio, César Muñoz, George Hagen, Anthony Narkawicz, and Swee Balachandran, *ICAROUS: Integrated Configurable Algorithms for Reliable Operations of Unmanned Systems*, Proceedings of the 35th Digital Avionics Systems Conference (DASC 2016), Sacramento, California, 2016

[2] Swee Balachandran, César Muñoz, María Consiglio, Marco Feliú and Anand Patel, *Independent Configurable Architecture for Reliable Operation of Unmanned Systems with Distributed On-Board Services*, Proceedings of the 37th Digital Avionics Systems Conference (DASC 2018), London, England, UK, 2018.

[3] Swee Balachandran, César Muñoz, and María Consiglio, *Distributed Consensus to Enable Merging and Spacing of UAS in an Urban Environment*, Proceedings of the 2018 International Conference on Unmanned Aircraft Systems (ICUAS 2018), Dallas, Texas, USA, 2018

[4] D. Ongaro and J. K. Ousterhout, "In search of an understandable consensus algorithm." in *USENIX Annual Technical Conference*, 2014, pp. 305–319.

[5] A. Koubâa, A. Allouch, M. Alajlan, Y. Javed, A. Belghith and M. Khalgui, "Micro Air Vehicle Link (MAVlink) in a Nutshell: A Survey," in *IEEE Access*, vol. 7, pp. 87658-87680, 2019.

[6] DJI, "Spreading Wings S1000," [Online]. Available: https://www.dji.com/spreading-wings-s1000. [Accessed 6 December 2019]

[7] Dronecode, "Cube Flight Controller," [Online]. Available: https://docs.px4.io/v1.9.0/en/flight_controller/pixhawk-2.html. [Accessed 6 December 2019].

[8] Intel, "Intel NUC mini pcs," [Online]. Available: https://www.intel.com/content/www/us/en/products/boards-kits/nuc.html [Accessed 23 April 2020].

[9] Verizon, "Verizon Jetpack MiFi," [Online]. Available: https://www.verizon.com/internet-devices/verizon-jetpack-mifi-7730l/#specsHeading. [Accessed 20 April 2020]

[10] AvaLAN, Ethernet to RF Module, [Online]. Available: https://cdn2.hubspot.net/hubfs/213677/AW900MTR_Product_Brief.pdf. [Accessed 20 April, 2020]