A Decentralized Framework to Support UAS Merging and Spacing Operations in Urban Canyons

Swee Balachandran[1], Christopher Manderino[2], César Muñoz[3] and María Consiglio[3]

***Abstract*— This paper presents a distributed consensus algorithm for autonomous merging and spacing that enables unmanned aircraft systems (UAS) to coordinate their passage through an aerial intersection (i.e., merging fix) via a distributed control mechanism. The algorithm is incorporated into the Independent Configurable Architecture for Reliable Operations of Unmanned Systems (ICAROUS). In-trail spacing between aircraft is achieved with the integration of sense and avoid functionality (SAA) within the ICAROUS framework. This approach allows vehicles to maintain required spacing while entering or exiting the merging fix. Enabling autonomously coordinating vehicles in merging and spacing operations is a key capability in facilitating vehicle traffic in urban airspace and air-mobility operations involving small UAS and other Electrical-Vertical Takeoff/Landing (E-VTOL) vehicles.**

I. Introduction

Several industries are exploring the use of small Unmanned Aircraft Systems (sUAS) to perform on-demand delivery of low weight, low cost consumer goods, mail, medical supplies, etc. Such delivery operations would result in a large number of vehicles operating in low altitude airspace when departing from and converging to distribution centers. The complexity of these operations and the variety of performance of these vehicles represent a technical challenge for dispatchers and schedulers of traffic management systems trying to prevent conflicts among converging flights while maximizing traffic flow. The features described in this paper will provide vehicles the capability to coordinate their arrival order and maintain in-trail spacing without intervention of a ground mission scheduler.

The emerging concept of operations known as Urban Air Mobility (UAM), led by the National Aeronautics and Space Administration (NASA), is set to revolutionize air travel. UAM, conceptually, relies on a dedicated network of airspace corridors that will accommodate a large number of concurrent flights. UAM operations are envisioned to have a high population of Vertical Takeoff/Landing (VTOL) vehicles operating in urban canyon type environments at any given time [1]. Vehicles converging to a vertiport need to maintain appropriate separation distance (i.e., spacing) with other vehicles while merging to the vertiport location. UAS Traffic Management (UTM) and UAM vehicles in intersecting and converging corridors will benefit from onboard technologies that ensure vehicle separation and spacing requirements.

Although offline entities such as UAS Service Suppliers (USS) can provide strategic deconfliction of flight plans a priori, the ability to autonomously perform merging and spacing operations in a decentralized fashion is required to dynamically respond to uncertainties associated with spacing or merging.

The goal of this paper is to explore the feasibility of performing merging and spacing operations solely relying on Vehicle to Vehicle (V2V) communications. This paper presents a merging approach that enables independent aerial systems to coordinate their passage through a common merging fix (i.e., the center of an intersection) in a distributed way. The merging algorithm is integrated into the Independent Configurable Architecture for Reliable Operations of Unmanned Systems (ICAROUS) framework and combined with its sense and avoid functionality. This merging algorithm allows vehicles to achieve and maintain required spacing when entering or exiting the merging fix. Furthermore, the presented algorithm also enables vehicles to transition from one merge fix to another. A proof of concept flight test demonstration of the algorithm presented in this work and associated flight test results are documented in [2].

This paper is organized as follows. Section II discusses related work. Section III discusses the underlying approach involving the scheduling algorithm and information exchange for achieving consensus. Section IV briefly explains how spacing between vehicles is achieved outside the intersection environment. Section V illustrates the proposed approach on a simple merging scenario in an urban environment. Section VI discusses various aspects of the proposed approach. Finally, Section VII offers conclusions and presents future work.

II. Related Work

Several systems are available to maintain separation from other vehicles in the airspace, e.g., TCAS [3], DAIDALUS [4], ACAS-X [5]. These systems are designed to provide resolution advisories to the pilot. The stability and control of intersecting aircraft flows were studied by Mao et al. [6].

Various techniques have been proposed to address merging and spacing problems for automotive vehicles at traffic inter-

[1]National Institute of Aerospace, Hampton, Virginia swee.balachandran@nianet.org

[2]University of Pittsburgh, Pittsburgh, Pennsylvania christopher.manderino@nsf-shrec.org

[3]NASA Langley Research Center, Hampton, Virginia {cesar.a.munoz,maria.c.consiglio}@nasa.gov

The research conducted in the paper was made possible through the Aeronautics Research Mission Directorate's Transformative Aeronautics Concepts Program and Formal Methods Group at NASA Langley Research Center; and, in part, by the industry and government members of the NSF SHREC Center and the IUCRC Program of the NSF under Grant No. CNS-1738783. The authors would also like to thank the operations and software assurance team at NASA Langley Research Center for their invaluable assistance with flight tests and software review.

sections. Scheduling-based approaches to coordinate vehicles approaching an intersection are used in [7]–[10]. Colombo et al. [11] use a schedule to construct a maximal control invariant set that will guarantee safe transit through an intersection. Zhang et al. [12]–[14] explore optimal control formulations where separation/collision constraints are represented as penalties in a cost function along with other constraints such as fuel consumption, ride smoothness, etc. The various approaches found in the literature can be classified into centralized vs. decentralized approaches. Centralized methods rely on a single common entity, often located at the intersection, to enable coordination. Decentralized approaches often decentralize the vehicle control problem while still relying on a centralized server to control sequencing and scheduling.

The approaches proposed by Zhang et al. [12], [14] could be used to control merging and spacing of UAS traffic in urban environments. However, these techniques would require a dedicated central agent for managing coordination and would lead to concerns in building a suitable infrastructure to host these dedicated agents. Likewise, a centralized approach must consider overhead in terms of economic impact from maintaining and operating the infrastructure. From a safety standpoint, a centralized solution may offer limited responses for autonomous vehicles to enter a state of graceful degradation before engaging fail-safe operations. A distributed consensus algorithm may be more resilient by allowing a system to mitigate individual unit failures.

This work explores a framework that enables vehicles to coordinate via V2V communication and reach consensus on a merging resolution determining when each vehicle can safely pass through an intersection. A V2V approach avoids the need for a dedicated, centralized coordinator that makes the decentralized approach an attractive solution for controlling the merging and spacing of UAS traffic in an urban environment.

III. MERGING

Given a set of predefined intersections and an arbitrary number of UAS approaching their merging fixes (i.e., intersections), the goal is to enable all vehicles to orderly traverse the intersection avoiding conflicts and collisions. In a centralized system, all agents approaching an intersection would communicate to a dedicated central agent residing at the intersection. The central agent coordinates the arrival/departure of each UAS approaching the intersection. In this paper, instead of having a dedicated central agent, each UAS approaching the intersection receives information about the arrival times of all other UAS approaching a shared intersection and computes a schedule such that its arrival at the intersection will ensure a safe spacing distance from other UAS.

This paper builds on preliminary work first discussed in [15]. Several improvements to the originally proposed merging solution is discussed in this work. Significant differences from the previous work is highlighted at relevant sections throughout this paper.

A. Scheduling Arrival Times

Given a set of n UASs approaching an intersection, let R_i, D_i denote the earliest and latest times, respectively, the i^{th} vehicle can approach the intersection. Let P denote the minimum separation time that must be maintained between vehicles crossing the intersection. The goal is to compute a schedule $T = (T_1, \ldots, T_n) \in \mathbf{R}^n$ for all $i \in \{1, \ldots, n\}$, such that

$$R_i \leq T_i \leq D_i - P \tag{1}$$

and thus, for all $i \neq j$

$$T_i \geq T_j \Rightarrow T_i \geq T_j + P. \tag{2}$$

Without loss of generality, the arrival times are normalized as follows:

$$r_i = \frac{R_i}{P}, \tag{3}$$

$$d_i = \frac{D_i}{P}. \tag{4}$$

A schedule $t = (t_1, \ldots, t_n)$ is computed with the normalized arrival times from (4) using a polynomial-time scheduling algorithm [11] as described in [15] and the crossing times for the original data is obtained as:

$$T_i = Pt_i,\ i \in \{1, \ldots, n\}. \tag{5}$$

1) Computing early arrival time: The earliest arrival time at the intersection is mathematically defined as:

$$R := \inf_{u \in \mathcal{U}} \{t : x(t, u, x(t_0)) = \mathbf{X}_{int}\}, \tag{6}$$

where $x(t, u, x(t_0))$ represents the position of the vehicle at time t when starting from initial condition $x(t_0)$ using the control input function $u \in \mathcal{U}$. The position $\mathbf{X}_{int}$ represents the intersection. Given the current time t_0, for a vehicle to reach the intersection from its current positions at the earliest time, it has to fly directly towards the intersection at its maximum speed:

$$R = t_0 + \frac{x_d - x_b}{v_{max}}. \tag{7}$$

Here, v_{max} represents the maximum speed of the vehicle, x_d represents the distance to the intersection. The value x_b represents a predetermined distance the vehicle is allowed to travel before actually computing a schedule. This value is chosen to provide sufficient leeway to compensate for factors such computational delays and network latency.

2) Computing late arrival time: Similar to the earliest arrival time calculation in Section III-A.1, a simple solution for the late arrival time is to use the slowest speed to fly directly towards the intersection. However, it is always desirable to maximize the latest arrival time at an intersection as this can be helpful in situations where there are multiple converging paths at an intersection and arrival times of vehicles are close together. The latest arrival time at the intersection is mathematically defined as:

$$D := \sup_{u \in \mathcal{U}} \{t : x(t, u, x(t_0)) = \mathbf{X}_{int}\}. \tag{8}$$

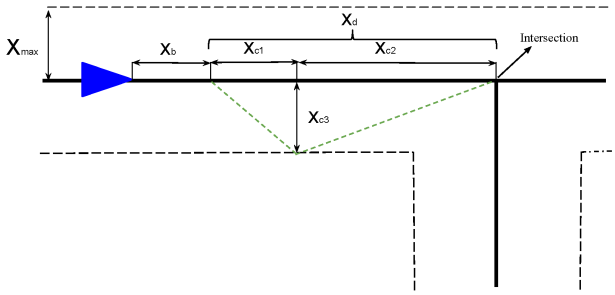

Fig. 1. Trajectories (in green) used by vehicle to maximize late arrival times

In this work, for simplicity, the class of inputs $\mathcal{U}$ is restricted to those that can yield trajectories of the form shown in Figure 1. The trajectories are parametrized by x_{c1} and x_{c3}. More specifically, each vehicle is allowed to make lateral deviations no greater than X_{max} from the nominal flight plan to maximize its late arrival time. Each vehicle is free to choose a suitable x_{c1}. Consequently, the late arrival time can be analytically computed as

$$D = t_0 + \frac{x_b}{v} + \frac{\sqrt{x_{c1}^2 + X_{max}^2} + \sqrt{(x_d - x_{c1})^2 + X_{max}^2}}{v_{min}}. \quad (9)$$

Here, v represents the current speed of the vehicle and v_{min}, which is assumed to be non-zero, represents the slowest speed possible for the vehicle.

3) Computing a trajectory given an arrival time: Once an arrival time t_a is computed by the scheduler, a suitable control function $u(t), t_0 \leq t \leq t_a$ that satisfies the following condition must be computed:

$$x(t_a, u, x(t_0)) = \mathbf{X}_{int}. \quad (10)$$

In general, the above constraint can be solved using an optimal control formulation [16]. Exploiting the class of trajectories used to find the late arrival time, a simpler solution can be obtained by searching for the cross track deviation $x_{c3} \in [0, X_{max}]$ and resolution speed $v_{res} \in [v_{min}, v_{max}]$ that satisfy the following equation:

$$\frac{x_b}{v} + \frac{\sqrt{x_{c1}^2 + x_{c3}^2} + \sqrt{(x_d - x_{c1})^2 + x_{c3}^2}}{v_{res}} = (t_a - t_0). \quad (11)$$

Note that Formula (11) is underdetermined and admits multiple solutions. Assuming x_{c1} is fixed a priori, one possible solution is to find a v_{res} that minimizes the lateral deviation x_{c3} (See Algorithm 2 in [15]).

B. Information Exchange and Consensus

Successfully computing a schedule, described in the previous section, requires each vehicle to use identical sets of predicted arrival times as each other vehicle that is approaching the same intersection. Let $\mathcal{V} = \{1, 2, ..., n\}$ represent the set of vehicles approaching a particular intersection. Let $(r_e, t_e, d_e)_{ij}$ represent vehicle i's knowledge of vehicle j, where $(r_e, t_e, d_e)_j$ represents the early, current and late arrival time information of vehicle j at epoch e, where, an epoch is a delineated passage of time.

A schedule computed by vehicle i, i.e., $(T_1, ..., T_n)_i$, is consistent with vehicle j's schedule if $(T_1, T_2, ..., T_n)_i = (T_1, T_2, ..., T_n)_j$. For vehicles to compute a consistent schedule, it is crucial that:

$$\forall\ i, j \in \mathcal{V} : (r_e, t_e, d_e)_{ij} = (r_e, t_e, d_e)_{jj} \quad (12)$$

i.e., there must be consensus among all vehicles on the flight data being used to compute a schedule, T.

A straightforward approach where vehicles broadcast their arrival data, i.e., (r_e, t_e, d_e) values, to every other vehicle in the network may not ensure the consistency condition described in Equation (12). For consistent and coherent flight data among vehicles, merging resolution uses a consensus-based approach.

In the preliminary version of this work presented in [15], the Raft algorithm enabled sUAS to achieve consensus. Generalizing the approach presented in [15] to networks of intersections challenges Raft's performance under real-time constraints. This work implements several modifications to the Raft algorithm, thus enabling the satisfaction of real-time and safety constraints presented by fixed-wing aircraft and rotorcraft. These modifications, however, eliminate some of Raft's guarantees that are not essential for the merging and spacing operations under real-time constraints. The adopted modifications are explained below.

To achieve consensus over arrival data, a virtual network $\mathcal{N}$ is established between the set of vehicles $\mathcal{V}$ approaching the intersection. Each vehicle in the intersection plays one of four roles: *witness, follower, candidate* or *leader*. The leader is responsible for sending *heartbeats*, M_k^{Hbt}, to followers at regular intervals to indicate the presence of a leader in the network and maintains its own liveness. Each epoch, e, is delineated by a leader heartbeat message M_k^{Hbt} where $k = e$.

If a leader is no longer live, leader accession must occur by another vehicle. Heartbeat messages contain information known to the leader– viz., intersection identifier, vehicles currently alive in the network, newly computed arrival times for vehicle approaching the intersection, and zone location of each vehicle (discussed in Section III-C).

Each vehicle in the network must receive a M_k^{Hbt} and send an acknowledgment message, M_k^{Ack}, to ensures its own liveness. Each M_k^{Ack} may encode a vehicle's computed arrival data (r_e, t_e, d_e). Consequently each vehicle is able to receive the arrival information about other vehicles in the network via this leader-follower heartbeat-acknowledgment process.

If a follower does not receive a heartbeat M_k^{Hbt} within a predefined timeout, a leader is considered failed. Follower vehicles enter the candidate state to requesting a minimum number of votes to enter the leader state. A Leader must first publish a heartbeat before safely resuming consensus-based merging operations. Additionally, any vehicle, irrespective of their current state, receiving a heartbeat from a leader of an intersection, immediately enters the follower state.

Maintaining consensus among vehicles on flight data is achieved as described above; however, in order to maintain safety, vehicles must agree on when merging schedules must be computed and executed. To facilitate consistent vehicle behavior, a specific structure is imposed on the volume of airspace around a merge fix and a set of rules are defined to govern merging and spacing operations.

C. Operational Rules During Merging

Considering an intersection to include the volume of airspace around a merge fix, an intersection is partitioned into three zones: Coordination zone, Scheduling zone and Entry zone, illustrated in Figure 2. Vehicles first enter the coordination zone, where a leader may first be identified; pass through the scheduling zone, sharing arrival times; and then traverse the entry zone before exiting via the merge fix. Note that the merge fix is uni-directional, i.e., it has a single exit direction. A vehicle first entering the coordination zone is a follower node. If a leader is not already identified, which is indicated by the absence of heartbeat messages, the follower transitions to a candidate and initiates the leader election process. Subsequent vehicles entering the coordination zones, become followers of the already established leader. The coordination zone is selected large enough to ensure vehicles have sufficient time to establish leadership and exchange arrival time information as described in the previous section.

A vehicle transitioning into the schedule zone from the coordination zone uses the arrival time information obtained from all vehicles in the intersection to check for separation conflicts and compute a new arrival schedule for the merge fix. The new arrival times computed by the new schedule are published to the leader for indexing and dissemination.

On ingress to the entry zone, a vehicle starts to execute the computed schedule in its log by choosing the appropriate control inputs to speed up or slow down/delay its arrival at the merge fix to meet the schedule constraints.

Vehicles in the entry zone are not allowed to make changes to their computed schedules and hence vehicles in the coordination and schedule zones must adjust their earliest arrival times to ensure they respect the schedule constraints of the vehicles that are ahead of them in the entry zone.

When a leader exits the merge fix airspace, the absence of heartbeat messages triggers follower vehicles in the intersection airspace to transition into the candidate state and initiate an election process to elect a new leader. On intersection egress, each vehicle switches into a neutral state (the witness state).

The incorporation of the operation rules involving the three zones discussed in this section is a modification to the preliminary work in [15]. These rules enable the information exchange to focus on achieving consensus on times of arrival. Consensus on when to compute and execute schedules are implicitly obtained by adhering to these rules of operations.

IV. Spacing

The scheduling process along with the operational rules described in the previous section ensure vehicles in the intersection airspace regulate their arrival times at the merge fix while maintaining sufficient separation/spacing between them. However, outside the intersection, vehicles maintain spacing with the sense and avoid functionality in ICAROUS. Sense and avoid capability in ICAROUS uses the DAIDALUS software library [4]. DAIDALUS provides horizontal (ground speed or track) and vertical (altitude or vertical speed) resolutions. Ground speed resolution doesn't require vehicles to deviate from their original path, hence it is a suitable candidate for maintaining spacing between vehicles approaching or departing from an intersection.

Fig. 2. A merge fix located in an urban environment. Various zones are indicated by the concentric rings

V. Results

As a proof of concept, the distributed merging and spacing algorithm is illustrated in a simulated scenario where three vehicles approach a merge fix in an urban environment, depicted in Figure 2. Figures 3-5 represent the vehicles' behaviors in the intersection. Each vehicle approaches the coordination zone with an initial speed of 7 m/s. As shown in Figure 5, the first vehicle to enter the coordination zone establishes itself as a leader and handles information exchange as described in section III-B. After entering the scheduling zone, all vehicles successfully compute a conflict free schedule that ensures a minimum separation of 10s at the merge fix. Figure 4 illustrates that the vehicles regulate their speed to satisfy the minimum separation criteria at the merge fix.

Figures 6-8 indicate a similar scenario where vehicle 2 and vehicle 3 enter the coordination zone 10s after vehicle 1. Consequently, vehicle 1 can reach the merge fix while maintaining sufficient separation from the other vehicles; here, vehicle 1 does not require a speed change. However, vehicles 2 and 3 compute schedules to ensure that they can achieve the required separation among themselves. Figure

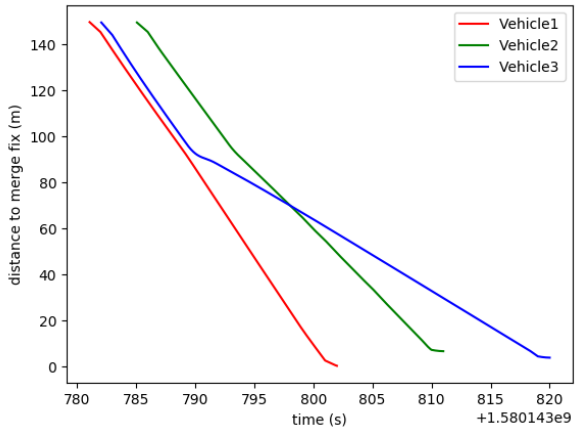

Fig. 3. Time vs Distance to merge fix

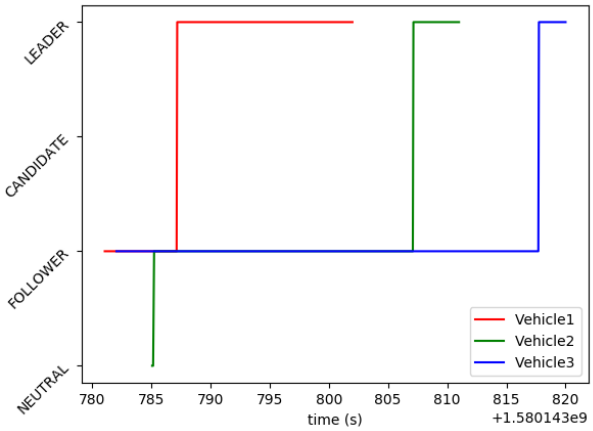

Fig. 5. Time vs Vehicle States

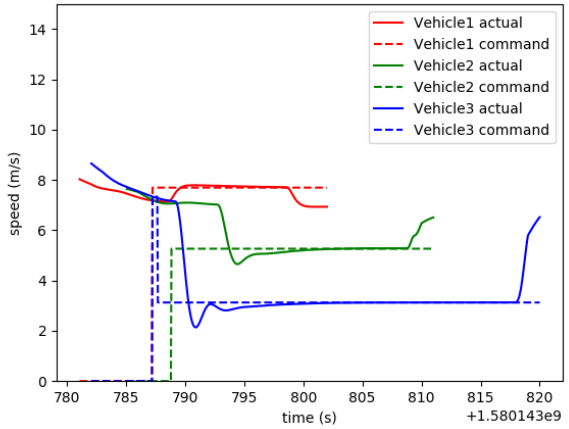

Fig. 4. Time vs Speed

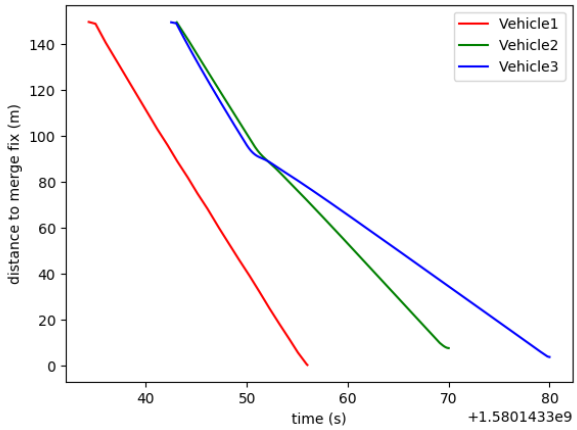

Fig. 6. Time vs Distance to merge fix

8 indicates the state transitions of the vehicle roles. Note that after vehicle 1 exits the intersection, vehicle 2 becomes the leader and ensures consistent data exchange among the vehicles currently in the intersection.

Flight test demonstrations and associated flight test results of the merging algorithm discussed in this work is presented in [2].

VI. Discussion

The intersection geometry consisting of the three zones proposed in this work serves as an implicit coordination mechanism which determine when a vehicle starts sharing information, computing a schedule, and executing its schedule. Consequently, the size of each of these zones influences the total number of UAS that can safely coordinate their passage through the intersection. For a given set of zone sizes, constraints on the maximum speed of vehicles passing through the airspace must be established to ensure that enough time is allowed for information exchange between vehicles to achieve consensus. Likewise, follower state time-

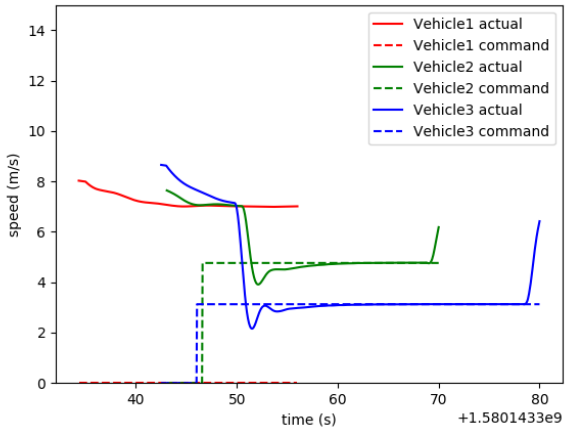

Fig. 7. Time vs Speed

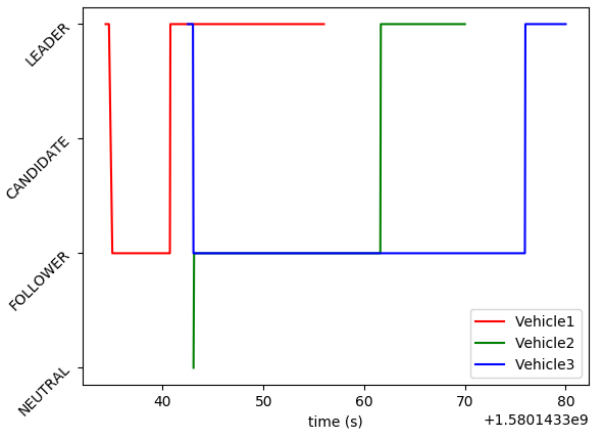

Fig. 8. Time vs Vehicle States

out parameters must be wisely chosen to reduce overhead when electing leaders.

V2V communication devices aboard each UAS are essential to: establish a network with other UASs participating in merging operations, enable information exchange, and achieve consensus. Currently available technologies such as Dedicated Short Range Communication (DSRC) and Cellular V2V technologies [17] are viable candidates.

Persistent network issues can degrade the performance of the system. In scenarios where link quality is significantly degraded, contingency maneuvers, such as loitering or hovering in a designated region of the airspace, are essential to ensure safe operation.

Path deviations, as a means to accommodate schedule solutions, are dependent on the geometry of the intersections. Extending trajectories with vertical maneuvers can accommodate a larger throughput at the intersection.

Preliminary simulation results demonstrated the application of the proposed work on three UAS vehicles. Future research directions will focus on studying the scalability of the proposed algorithm and analyzing its limitations in terms of the number of vehicles that can be handled for a given volume of airspace. These studies will be crucial for identifying requirements, limitations and restrictions for multi vehicle merging and spacing operations. The operational rules discussed in this paper will have to account for cases where vehicles fail to achieve consensus due to factors such as communication issues and onboard emergencies.

VII. Conclusion

This paper proposes a framework where merging and spacing constraints between vehicles can be satisfied using a combination of scheduling and distributed consensus integrated with the sense and avoid capability of ICAROUS. Vehicles approaching an intersection form a network of nodes, each playing a specific role. The vehicle nodes elect a leader that helps collate information across all vehicles in the network. Consequently, each vehicle computes the same solution, enabling them to validate agreement on their crossing times. A vehicle leaves the network once it has safely crossed the intersection. New vehicles approaching the intersection become new members of the network. When leaders drops out, a new leader is automatically elected, enabling existing members and new members to coordinate safe passage through the intersection.

In summary, the proposed framework demonstrates the viability of coordinating safe passage through an intersection in a decentralized manner. Future work will focus on analyzing the safety properties of this framework, including identifying initial conditions when a solution to the scheduling problem may not exist, describing possible resolutions to deal with such conditions, and evaluating the system's ability to tolerate network latency and communication failures.

References

[1] D. P. Thipphavong, R. Apaza, B. Barmore, V. Battiste, B. Burian, Q. Dao, M. Feary, S. Go, K. H. Goodrich, J. Homola *et al.*, "Urban air mobility airspace integration concepts and considerations," in *2018 Aviation Technology, Integration, and Operations Conference*, 2018, p. 3676.

[2] A. Peters, S. Balachandran, B. Duffy, K. Smalling, M. Consiglio, and C. Muñoz, "Flight test results of a distributed merging algorithm for autonomous uas operations," in *Digital Avionics Systems Conference (DASC), 2020 IEEE/AIAA 39th*. IEEE, 2020, p. (to appear).

[3] T. Williamson and N. A. Spencer, "Development and operation of the traffic alert and collision avoidance system (tcas)," *Proceedings of the IEEE*, vol. 77, no. 11, pp. 1735–1744, 1989.

[4] C. Muñoz, A. Narkawicz, G. Hagen, J. Upchurch, A. Dutle, and M. Consiglio, "DAIDALUS: Detect and Avoid Alerting Logic for Unmanned Systems," in *Proceedings of the 34th Digital Avionics Systems Conference (DASC 2015)*, Prague, Czech Republic, September 2015.

[5] M. J. Kochenderfer, J. E. Holland, and J. P. Chryssanthacopoulos, "Next generation airborne collision avoidance system," *Lincoln Laboratory Journal*, vol. 19, no. 1, pp. 17–33, 2012.

[6] Z.-H. Mao, E. Feron, and K. Bilimoria, "Stability of intersecting aircraft flows under decentralized conflict avoidance rules," in *18th Applied Aerodynamics Conference*, 2000, p. 4271.

[7] K. Dresner and P. Stone, "Multiagent traffic management: A reservation-based intersection control mechanism," in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*. IEEE Computer Society, 2004, pp. 530–537.

[8] ——, "Multiagent traffic management: An improved intersection control mechanism," in *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*. ACM, 2005, pp. 471–477.

[9] L. Bruni, A. Colombo, and D. Del Vecchio, "Robust multi-agent collision avoidance through scheduling," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE, 2013, pp. 3944–3950.

[10] D. Miculescu and S. Karaman, "Polling-systems-based autonomous vehicle coordination in traffic intersections with no traffic signals," *arXiv preprint arXiv:1607.07896*, 2016.

[11] A. Colombo and D. Del Vecchio, "Efficient algorithms for collision avoidance at intersections," in *Proceedings of the 15th ACM international conference on Hybrid Systems: Computation and Control*. ACM, 2012, pp. 145–154.

[12] Y. J. Zhang, A. A. Malikopoulos, and C. G. Cassandras, "Optimal control and coordination of connected and automated vehicles at urban traffic intersections," in *American Control Conference (ACC), 2016*. IEEE, 2016, pp. 6227–6232.

[13] L. Zhao, A. Malikopoulos, and J. Rios-Torres, "Optimal control of connected and automated vehicles at roundabouts: An investigation in a mixed-traffic environment," *arXiv preprint arXiv:1710.11295*, 2017.

[14] J. Lee and B. Park, "Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 1, pp. 81–90, 2012.

[15] S. Balachandran, C. Muñoz, and M. Consiglio, "Distributed consensus to enable merging and spacing of uas in an urban environment," in *2018 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2018, pp. 670–675.

[16] D. E. Kirk, *Optimal Control Theory, An Introduction*. Prentice-Hall, Inc., 1970.

[17] K. Abboud, H. A. Omar, and W. Zhuang, "Interworking of dsrc and cellular network technologies for v2x communications: A survey," *IEEE transactions on vehicular technology*, vol. 65, no. 12, pp. 9457–9470, 2016.