Distributed Consensus to Enable Merging and Spacing of UAS in an Urban Environment

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

***Abstract*— This paper presents a novel approach to enable multiple Unmanned Aerial Systems approaching a common intersection to independently schedule their arrival time while maintaining a safe separation. Aircraft merging at a common intersection are grouped into a network and each aircraft broadcasts its arrival time interval to the network. A distributed consensus algorithm elects a leader among the aircraft approaching the intersection and helps synchronize the information received by each aircraft. The consensus algorithm ensures that each aircraft computes a schedule with the same input information. The elected leader also dictates when a schedule must be computed, which may be triggered when a new aircraft joins the network. Preliminary results illustrating the collaborative behavior of the vehicles are presented.**

I. Introduction

Recent advances in sensor and vehicle technologies have sparked an interest in the development of air transportation systems for Urban Air Mobility. These systems satisfy on-demand mobility needs in densely populated cities where ground transportation is very time consuming. Some of these transportation systems assume that the urban airspace will be shared with small Unmanned Aerial Systems (sUAS) that perform door to door package delivery services, for example. Although these promising concepts are still in their inception, their realization requires the understanding of the challenges associated with air traffic management in the urban airspace.

Air traffic management applied to manned aviation (commercial and general aviation) has been an active field of research over the past decades. Operations en-route and in terminal environments have been studied in depth and consequently several algorithms to optimize traffic flow and minimize conflicts have been proposed. While air traffic operations, both manned and unmanned, in controlled airspace are strictly regulated, sUAS operating in uncontrolled airspace will require new and different traffic management methods. In an urban environment, maintaining constant line of sight or ground-based radar contact with these vehicles is impossible due to obstruction from buildings. This work proposes a multiagent coordination framework to regulate the flow of air traffic in urban environments. More specifically, this paper explores problems arising in merging, sequencing, and coordination in an urban airspace.

Figure 1 illustrates an urban setting where several users of the airspace utilize common airways to accomplish individual missions. The main contribution of this work is a novel approach that allows multiple aerial vehicles to safely merge at a common intersection. These vehicles coordinate passage through the intersection by independently scheduling their arrival times, while avoiding loss of separation. Vehicles approaching the intersection use the Raft consensus algorithm [1] to initiate a network. The vehicles elect a leader using Raft's voting mechanism. Each aircraft broadcasts an arrival time interval to the leader. The leader ensures that this information is synchronized across all vehicles in the network by virtue of Raft's log replication property. The leader also monitors for imminent loss of separation at intersections and dictates when a schedule must be computed to prevent a conflict. Each aircraft computes its individual arrival time on receiving the appropriate log entry from the leader. The proposed approach can accommodate new aircraft into the network. When necessary, the leader prompts every aircraft in the network to reschedule arrival times. Preliminary simulation results illustrating the collaborative behavior of the vehicles are presented.

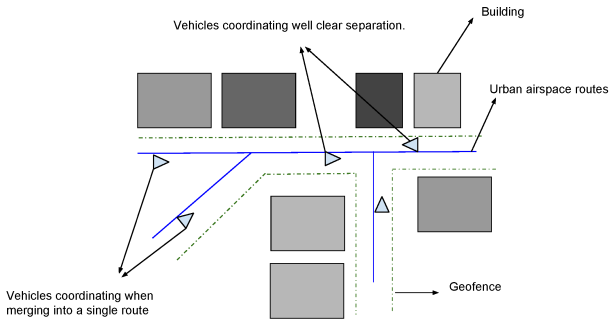

Fig. 1. UAS operation in an urban environment involving specific air routes. These air routes facilitate traffic flow to accommodate various missions.

This paper is organized as follows. Section II discusses related work. Section III provides background information on the proposed scheduler and the Raft consensus algorithm. Section IV and V outline the problem addressed by this paper and provides a detailed description of the proposed solution. Section VII illustrates results with use case examples. Section VIII discusses various aspects of the proposed approach. Finally, Section IX concludes and presents future work.

II. Related Work

Collision avoidance for manned aircraft and detect and avoid technologies for Unmanned Aerial Systems (UAS) have led to the development of systems such as TCAS

[1]National Institute of Aerospace, Hampton, Virginia 23666 swee.balachandran@nianet.org

[2]NASA Langley Research Center, Hampton, Virginia 23666 {cesar.a.munoz,maria.c.consiglio}@nasa.gov

[2], ACAS-X [3] and DAIDALUS [4]. These systems are designed to provide resolution advisories to pilots. Mao et al. [5] studied the stability and control of intersecting aircraft flows in high altitude airspace.

With the advent of automotive Vehicle to Vehicle (V2V), Vehicle to Infrastructure (V2I) technologies and an increasing interest in self-driving vehicles, the problem of coordinating vehicles entering and leaving an intersection has gained a lot of traction among automotive researchers. Various techniques have been proposed to address the problem of merging and spacing of vehicles at traffic intersections. Scheduling based approaches to coordinate the arrival times of vehicles approaching an intersection was used in [6]–[9]. Colombo et al. [10] used the schedule to construct a maximal control invariant set that will guarantee safe passage of vehicles through an intersection. Zhang et al. [11]–[13] explore optimal control formulations where separation/collision constraints are represented as penalties in a cost function along with various other constraints such as fuel consumption, ride smoothness, etc. The approaches found in the literature can be roughly classified into centralized vs. decentralized approaches. The centralized methods rely on a central entity, often located at the intersection, to enable coordination. The decentralized approaches often decentralize the vehicle control problem while still relying on a centralized server to control sequencing/scheduling.

The techniques in [11], [13] could potentially be used to control merging/spacing of UAS traffic flows in urban environments. However, these techniques would require a dedicated central agent for managing coordination. This would lead to concerns such as building a suitable infrastructure to host these central agents, along with the economic impact of maintaining and operating this infrastructure. From a safety standpoint a centralized solution suffers from having a single point of failure. A distributed consensus algorithm may be more resilient if appropriate mitigations for individual failures are considered.

This work explores a framework that enables vehicles to coordinate via a communication link and come to a consensus on when each vehicle can safely pass through the intersection. This approach avoids the need for a dedicated centralized coordinator thus making it an attractive solution for controlling merging and spacing of UAS traffic in an urban environment.

III. Background

A. Scheduling

Given a set of jobs and a set of resources needed to complete the jobs, scheduling is the decision making process in which jobs are assigned to resources at specific execution times [14]. The scheduling problem or the job-shop problem is represented by the tuple (α, β, γ), where the element α describes the machine environment (e.g., total number of machines), the element β defines the jobs characteristics (e.g., job duration, precedence constraints), and the element γ defines the optimality criterion (e.g., makespan).

Without loss of generality, this paper focuses on the scheduling problem defined by $\alpha = 1$, $\beta = (r_i, d_i, p_i)$ and $\gamma = max_i(t_i + p_i - d_i)$. Here, the times r_i, d_i, p_i represent the earliest possible start time (release time), the deadline, and the duration of the i^{th} job, respectively. The time t_i represents the scheduled start time of the i^{th} job. The translation of determining safe passage through an intersection to the above scheduling problem is described in detail in Section V.

B. The Raft Consensus Algorithm

Raft [1] is a consensus algorithm that enables replicating values across multiples nodes participating in a network. The set of values to be replicated across different nodes are collectively referred to as the *log*. There are two fundamental components to the Raft algorithm: leader election and log replication. Raft achieves consensus by first electing a node as a leader. The leader assumes the responsibility of managing the replicated log. The leader accepts log entries from clients and replicates them on other nodes and also tells them when it is safe to make use of these values for computation. The leader node first adds a new entry to its log and data transfer always takes place from the leader to other nodes in the network. The Raft algorithm can accommodate leader failures (e.g., leader disconnects from the network) by electing new leaders to manage the network.

Each node in the cluster can exist in one of three states: follower, candidate, and leader. Nodes start as followers and expect to receive regular heartbeats from a leader node. If a follower does not receive a heartbeat within a time interval, it transitions to a candidate and triggers an election process governed by random timeouts. Followers vote for a candidate and the candidate with the majority votes becomes the leader. A new leader is elected when starting the cluster and when an existing leader fails.

The Raft algorithm has the following important safety property [1]: if any node has used a particular log entry (e.g., to perform computations), then no other node may use a different log entry for the same log index.

IV. Problem Statement and Approach

Given a set of predefined intersecting paths and an arbitrary number of UAS traversing these paths, the goal is to enable all vehicles to orderly cross 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. Instead of having a dedicated coordinating agent, in this work, each UAS approaching the intersection receives information about the crossing times of all other UAS approaching the intersection and computes a schedule such that its arrival at the intersection will ensure a safe spacing distance from other UAS. One potential problem with a naive implementation of the proposed approach is that it is crucial to ensure that all agents agree on the schedules they compute individually. In other words, each

agent must use the exact same input information to compute the schedule. To resolve this issue, the Raft consensus algorithm is used to enable synchronization of values across all vehicles thus ensuring each vehicle uses the same set of information to compute a schedule. Section V illustrates how the scheduler constructs the optimal arrival times. Section VI explains how the network of UASs use the Raft consensus algorithm to ensure consistent schedule computation.

V. Scheduling Arrival Times

A. Schedule Construction

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 $i \neq j$

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

In scheduling terms, the above problem is of the form $\alpha = 1, \beta = (r_i = R_i, d_i = D_i, p_i = P), \gamma = L_{max} : max_i(T_i + P_i - D_i)$. Without loss of generality, one can consider the case where $p_i = 1$ by normalizing the data as follows:

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

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

The schedule $t = (t_1, \ldots, t_n)$ is computed using Algorithm 1 [10] as described in the appendix and the crossing times for the original data is obtained as:

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

B. 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.

C. Computing late arrival time

Similar to the earliest arrival time calculation in Section V-B, 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:

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

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 2. 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

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

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.

D. 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:

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

In general, the above constraint can be solved using an optimal control formulation [15]. 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:

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

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).

VI. Synchronizing Information Exchange using Raft

A. Raft cluster initiation and membership changes

Each UAS approaching an intersection, by default, starts in a *neutral* state. Each UAS scans to check if it receives a heartbeat message from the Raft leader managing the intersection. If a network of Raft servers is not already available, the first UAS that is approaching the intersection

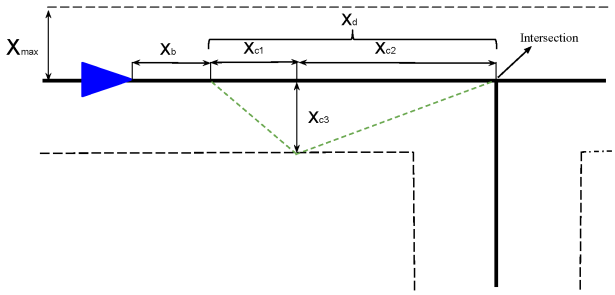

Fig. 2. Trajectories (in green) used by vehicle to maximize late arrival times

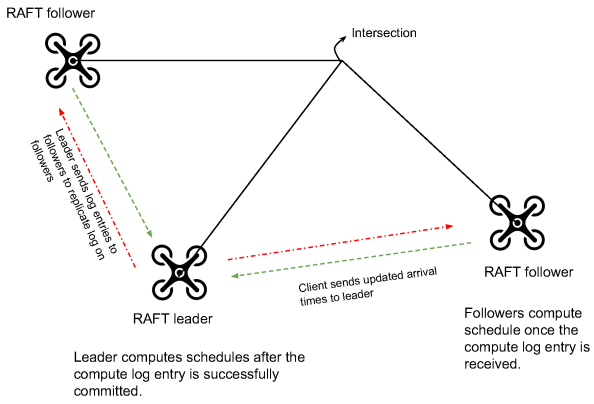

Fig. 3. Schematic representation of data transfer between nodes in a Raft network among UAS approaching an intersection.

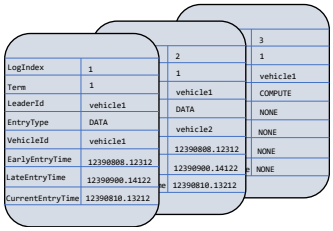

Fig. 4. Typical accumulation of log entries during a merging operation between two vehicles

and is closest to the intersection becomes a leader. If two or more aircraft are the same distance away, a tie-breaking technique, for example based on the aircraft identifiers, can be considered. Once a leader is established, other aircraft can join the network by requesting membership from the leader. Addition and removal of nodes to the network is facilitated by the Raft algorithm. Once a UAS safely passes through the intersection, it drops off from the network and transitions to a *neutral* state. If a leader drops off the network, a new leader is elected among the existing nodes in the network by virtue of the Raft algorithm.

B. Log replication

Figure 4 illustrates a typical log sequence. A `DATA` entry consists of information about the Raft network (e.g., log index, leader, term) and the arrival times of a specific aircraft. Each vehicle computes its early, late, and current arrival times and broadcasts this information to the leader of the Raft network. The leader creates a log entry with the received information and ensures that this entry is consistently replicated in the logs of other follower nodes in the network. The leader also constantly monitors the received information to ensure each vehicle passing through the intersection maintains a safe separation from each other. If the leader detects an imminent conflict due to loss of separation, it appends the `COMPUTE` entry to the log and replicates it across other members. Once a follower receives a compute entry, it starts computing a schedule. The leader starts computing a schedule once the command entry is successfully replicated across a majority of the nodes in the network.

VII. Results

A prototype of the proposed decentralized scheduling approach was implemented and a simple scenario of three vehicles approaching an intersection, as shown in Figure 3, was simulated. The initial conditions of the three vehicles were picked such that in the absence of any flight plan changes, the three vehicles would be collocated at the intersection thus causing a loss of separation among the three vehicles. Figure 5 illustrates the distance to the intersection of each vehicle with the proposed framework. As these vehicles approach the intersection, the three vehicles form a Raft cluster and elect a leader (see Figure 6). The leader synchronizes all their logs ensuring each vehicle has information about the earliest, late, and current crossing times of other vehicles at the intersection. The leader detects that the current crossing times of all the vehicles result in a loss of separation and appends a `COMPUTE` schedule command to each vehicles log. On receiving the command, each vehicle computes a schedule and executes a trajectory consistent with its schedule as described in Section V-D. In this example, the required minimum separation time at the intersection was chosen to be $10s$. As seen if Figure 5, each vehicle makes adequate changes to its trajectory such that it crosses the intersection while satisfying separation constraints.

VIII. Discussion

This work assumes the existence of a vehicle to vehicle (V2V) communication device aboard each UAS to establish a network and exchange information. Currently available technologies such as Dedicated Short Range Communication (DSRC) and Cellular V2V technologies [16] are viable candidates. Note that the proposed framework provides some leeway x_b (see section V-B) for the vehicle before executing the computed schedule. This provides a certain amount of robustness against network latency and intermittent packet dropouts. However, persistent network issues can degrade the performance of the system. When a new member is added to the Raft network at an intersection, the leader ensures that the new member's log catches up with the most updated log.

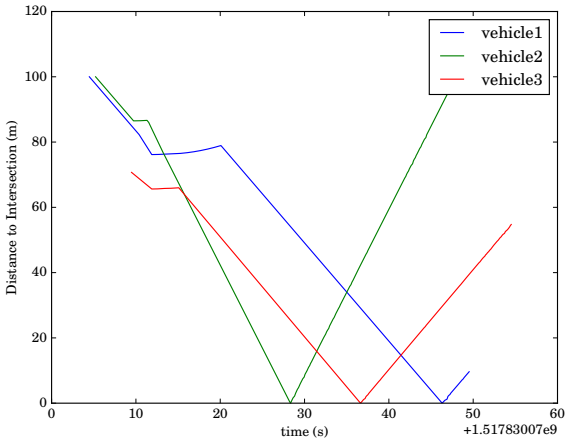

Fig. 5. Distance to intersection: Each aircraft deviates from its nominal trajectory to meet the computed schedule constraints

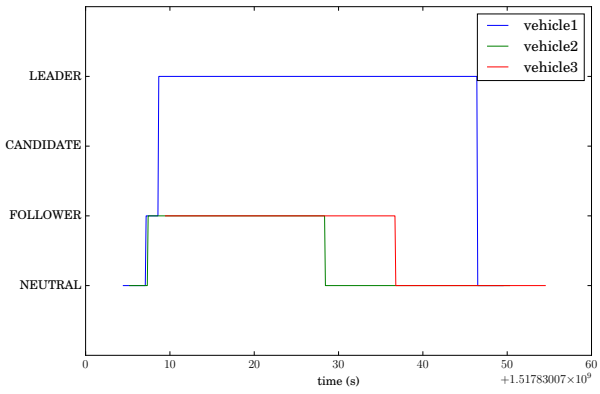

Fig. 6. Progression of Raft states as aircraft fly through the intersection

This can result in follower logs being populated with stale data that is no longer necessary for schedule computation. This can be eliminated by using suitable log compaction strategies [1]. The path deviations to accommodate solutions is dependent on the geometry of the intersections. Extending trajectories with vertical maneuvers can accommodate a larger throughput at the intersection.

IX. Conclusion

This paper proposed a framework where merging and spacing constraints between vehicles can be satisfied using a combination of scheduling and distributed consensus. The vehicles approaching an intersection form a Raft network and elect a leader. The leader of the network helps synchronize the same set of information across all vehicles in network. Consequently, each vehicle computes the same solution, enabling them to agree on the crossing times of each other. A vehicle leaves the network once it has safely crossed the intersection. New vehicles approaching the intersection become new members of the network. If a leader drops out, a new leader is elected by the Raft consensus algorithm thus 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. More specifically, identifying initial conditions when a solution to the scheduling problem may not exist and possible resolutions to deal with such conditions and analysis on the robustness of the system to network latency and communication failures.

Appendix

Algorithm 1 Scheduling algorithm (From [10])

Require: $r = \{r_1, \ldots, r_n\}$ release times of jobs,
$d = \{d_1, \ldots, d_n\}$ deadlines of jobs
for all $i \in \{i, \ldots, n\}$ **do**
  $F_i \leftarrow \emptyset$, $c_i \leftarrow \emptyset$
end for
sort jobs in increasing r_i: $(r_1 \leq r_2, \ldots, r_n)$
for $i = n$ downto 1 **do**
  for all $j \in \{i, \ldots, n\}$ such that $d_j \geq d_i$ **do**
    if $c_j = \emptyset$ **then**
      $c_j \leftarrow d_j - 1$
    else
      $c_j \leftarrow c_j - 1$
    end if
    while $c_j \in F_k$ for some F_k **do**
      $c_j \leftarrow \inf(F_k)$
    end while
  end for
  if $i = 1$ or $r_{i1} < r_i$ **then**
    $c \leftarrow \min_i(c_i)$
    if $c < r_i$ **then**
      return $\emptyset$
    end if
    if $c \in [r_i, r_{i+1}]$ **then**
      $F_i \leftarrow [c - 1, r_i]$
    end if
  end if
end for
$t \leftarrow 0$
$T_i = 0, \forall i \in \{1, \ldots, n\}$
for $i = 1$ to n **do**
  $r_{min} \leftarrow \min\{r_j$: job j has not been scheduled$\}$
  $t \leftarrow \max\{t, r_{min}\}$
  while $t \in F_j$ for some j **do**
    $t \leftarrow \sup(F_j)$
  end while
  $j \leftarrow \{i$: job i has least d_i among jobs ready at t $\}$
  $T_j \leftarrow t$
  $t \leftarrow t + 1$
end for
return T

Algorithm 2 Computing v_{res} and x_{c3} given t_a

Require: $x_{c1}, x_d, t_a, \delta_x$

for $x_{c3} \in [0, \delta_x, 2\delta_x, \ldots, X_{max}]$ **do**
  $v_{res} = \frac{\sqrt{x_{c1}^2+x_{c3}^2}+\sqrt{(x_d-x_{c1})^2+x_{c3}^2}}{t_a}$
  if $v_{min} \leq v_{res} \leq v_{max}$ **then**
    return v_{res}, x_{c3}
  end if
end for

REFERENCES

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

[2] 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.

[3] 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.

[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] 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.

[6] 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.

[7] ——, "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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] M. L. Pinedo, *Scheduling: theory, algorithms, and systems*. Springer, 2016.

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

[16] 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.