Guaranteeing Partial State Synchronization for UAV Platoon under Message or UAV Loss

Baisakhi Chatterjee
Computer Science Department
North Carolina State University
bchatte@ncsu.edu

Rudra Dutta
Computer Science Department
North Carolina State University
rdutta@ncsu.edu

***Abstract*—Unmanned Aerial Vehicles (UAVs) are increasingly being used to perform tasks either individually, or in groups. As more and more applications of UAV platoons are emerging, it is becoming apparent that coordination based on dynamically sensed information is critical for mission success. In this paper, we have addressed the concerns of fallible communication and an unreliable system on UAV platoon coordination. We have studied how robustness may be designed in a multi-agent scenario with arbitrary number of agents and developed an algorithm which guarantees distributed partial state synchronization with bounded variance in number of states despite facing message loss or system failure in agents. Our algorithm allows agents to autonomously take corrective actions in order to ensure the system remains stable. Results show that our algorithm is efficient, scalable and highly capable of achieving these goals.**

***Index Terms*—unmanned aerial vehicle, autonomous agents, distributed synchronization, multiparty state synchronization,**

I. Introduction

In recent times, unmanned aerial vehicles have experienced a surge in popularity due to the versatility of their usage. UAV platoons can efficiently perform a variety of tasks in hazardous conditions without human intervention. We use the term "platoon" to indicate a group of UAVs that share a common purpose (a "mission") which represents the intents of a single stakeholder, such that the mission requires various UAVs of the group to perform related but different tasks, requiring trajectory coordination. Examples of platoon applications can range from highly distributed operations such Search-and-Rescue [1], agricultural monitoring [2] and target tracking [3], to more localized ones such as Plume Wrapping [4].

As is obvious, each UAV in a platoon of autonomous UAVs must, at a minimum, exchange location information on an ongoing basis with some or all of the rest of the platoon. This enables the other UAVs to make trajectory decisions autonomously, while collectively ensuring that the goals of a mission related to distribution, coverage, etc. be satisfied. Wireless communication, however, is subject to message delay and loss. It is further possible that UAVs experience system failure [5] which force them to abort the mission. In these situations, it is imperative that the UAV platoon detects this failure and is able to take steps to either synchronize and continue with the task, or abort the mission safely. Our goal in this work is, thus, to facilitate communication in a mission-oriented multi-UAV platoon where reliability is not guaranteed. To do this, we propose an approach for preserving bounded multiparty synchronization in an autonomous distributed system and choose the task of Plume Wrapping as an exemplar. Plume Wrapping, which we take as a running example in the rest of this paper, is the problem of mapping the boundary of a potentially dangerous airborne material, such as gas or toxic leak. Such leaks may be a result of deliberate malicious activity, or simple accidents [6]. Mapping the extent of the plume allows emergency responders to collect important information about the situation [7]. A fully autonomous group of UAVs can fly into such unsafe conditions and coordinate with each other to successfully surround the plume. Each agent collects local information which must be transmitted to the platoon such that the group can plan their individual trajectories based on the collective information.

In our earlier work [8], we developed a Plume Wrapping algorithm that allows UAVs to make independent decisions based on input from other agents in the group. We modeled message loss implicitly, as message delay incurred due to retransmission of messages. In this paper, we have explicitly dealt with message loss where information has been lost and retransmission is required for UAVs to successfully complete their task. We have further introduced the possibility of UAV system failure where one or more agents may be unable to transmit or receive messages and the platoon must take corrective action to continue the mission. It is important to note, while we focus on the mission of Plume Wrapping, our algorithm can be applied to synchronize coordination in UAV platoons in general, simply by altering the mission specifications. Our results show that it is possible to guarantee partial distributed multiparty synchronization and ensure a task is completed despite operating under less than ideal circumstances.

II. Relevant Prior Work

Communication and coordination in autonomous agents have been studied extensively in a wide variety of applications. In target tracking, for example, multiple autonomous UAVs can share their local observation data in order to take actions based on the shared information [9], [10]. Multi-UAV

The material presented in this paper is based upon work supported by the National Science Foundation PAWR Program under Grant No.: CNS-1939334, and its supplement for studying NRDZs.

platoons have been employed for agricultural monitoring [11], surveillance [12] and disaster management [13]. Considerable research has also been conducted in the field of plume localization and tracing. Fu et al. [14] proposed an odour based localization algorithm to determine the source of a pollution emission where UAVs planned their path by sharing local observational data with the group to plan their trajectory. Zarzhitsky et al. [15] designed a model-based Plume Wrapping algorithm using only local information observed by the agents. In [4], Babu and Dutta demonstrated how a group of UAVs could wrap a plume by sharing positional updates. In all of these works, agents were required to exchange information with each other to successfully complete the mission. However, all the authors have considered an idealized system where no information loss or system failure occurred. A real-world scenario consists of both of these challenges and, as such, it is important to consider scenarios where such issues occur.

Failure detection and mitigation of a coordinated group of autonomous agents has been studied widely in multi-UAV groups [16], [17]. Both works considered a perfectly reliable channel and a single UAV failure. Huang et. al considered restarting the entire algorithm in case of loss [18]. In our earlier work [8], we designed a Plume Wrapping algorithm based on the Fibonacci Spiral Method. While we considered message delay due to packet loss, we did not explicitly design or study this scenario. In this paper, we have analyzed the effects of message loss on mission completion and correctness and designed an algorithm to mitigate its effects. We have further studied UAV loss and proposed an algorithm that enables the UAVs to continue the mission despite the loss of multiple members of the platoon, thus guaranteeing preservation of distributed multiparty synchronization.

III. Our Problem Model

The goal of our research is to show that we can guarantee partial multiparty synchronization in a distributed autonomous system despite presence of an unreliable channel and an imperfect system when pursuing any mission that requires trajectory coordination and ongoing location information exchange in a UAV platoon. The problem can be generalized to any UAV platoon mission where members of the group require updated knowledge of other agents' states. Our overarching aim is to preserve bounded multiparty state synchronization for any group of agents completing a mission in a distributed fashion with a bounded range of states and monotonic increase of states for each individual member. In case of operations like Search-and-Rescue, agents may use updated location and survivor information to plot trajectory. For Plume Wrapping, agents use collective density information to make decisions. While the choice of Plume Wrapping was motivated by the both practical considerations and existing academic interest, we only use this problem as an exemplar. In this section, we first briefly describe our prior work on Plume Wrapping.

A. Prior Plume Wrapping Approach

In [8], we designed an algorithm which could autonomously surround a spherical plume by exchanging only position and plume density information with the group. To do this, we reduced this problem to that of placing n points on a sphere using the Fibonacci Spiral method. The choice of a spherical sphere was motivated by the fact that the starting phase of a plume with pollutants lighter than air will assume a roughly spherical shape [19]. Since we are focused exclusively on achieving synchronization during communication, we have opted for a simplified example of the Plume Wrapping problem.

Initially, a single UAV flies in an arbitrary direction to reach the plume boundary, after which other members of the platoon calculate target positions on the surface of a hypothetical sphere near the boundary and move to occupy this position. Once the N^{th} UAV has finished moving to the target position on the surface of the imaginary sphere, the algorithm proceeds to a partially synchronized stage called the *Stop-Click Phase*. In this phase, each UAV exchanges messages containing state information which, in our Plume Wrapping example, contain position and density information. The UAVs then wait to receive $N - 1$ updates from other agents. Based on the values observed, the UAVs make trajectory planning decisions before moving to the next stage. In Plume Wrapping, UAVs choose either *Expansion* (if all agents are inside the plume boundary) or *Rotation* (when at least one agent is outside the boundary). During *Expansion*, the UAV increases the radius of the imaginary sphere and then recalculates its target position on it. For *Rotation*, the UAVs calculate the target position such that the imaginary UAV sphere pivots around the anchor and moves inwards towards the plume. Though each UAV makes this choice independently, the group as a whole makes the same choice since they operate on the same data. Once the choice is made, each UAV moves to the target position and broadcasts its current position and density. For message exchange, we adopted an abstraction of the communication medium with no message loss and bounded message delay. This process is repeated iteratively till all the UAVs reach the plume boundary.

It is important to note here that '*Rotation*' and '*Expansion*' are exclusive to the problem of Plume Wrapping. Our algorithm utilized mission-specific data detected at each UAV's current position for a decision based iterative adaptive trajectory control. Depending on the data detected, each UAV would make a choice on its movement. Plume Wrapping, for example, requires position and plume concentration updates for the UAVs to plan their paths. A limitation, however, of our previous work was that it cannot guarantee synchronization in the face of message loss. If some message from UAV$_i$ to some other UAV$_j$ was dropped by the channel, UAV$_j$ might make a decision on incorrect data and thus lose synchronization. Alternatively, if some UAV$_k$ suffered a failure such that it was unable to participate in the algorithm, this would cause the entire platoon to stall. An example of this limitation is

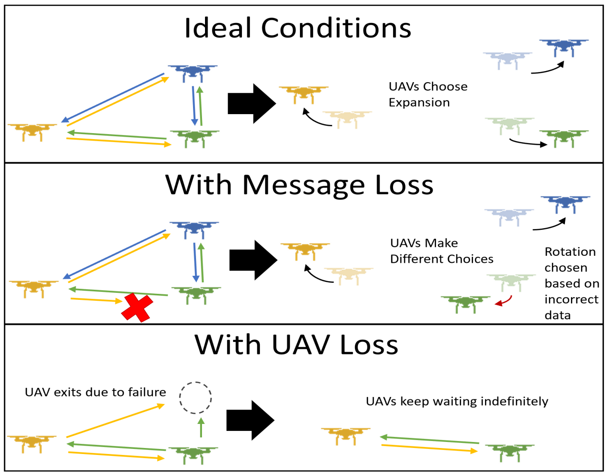
Ideal Conditions
UAVs Choose Expansion
With Message Loss
UAVs Make Different Choices
Rotation chosen based on incorrect data
With UAV Loss
UAV exits due to failure
UAVs keep waiting indefinitely

Fig. 1. Behaviour of Original Algorithm in (a) Ideal Conditions, (b) Message Loss and (c) UAV Failure Scenarios. In case of message loss, the UAVs lose synchronization. In case of UAV loss, the algorithm stalls.

shown with 3 UAVs in Fig. 1. In this work we have attempted to solve this problem by allowing UAVs to take corrective actions when such scenarios occur.

IV. Multiparty Synchronization Algorithm

In this section, we describe our algorithm to guarantee partial synchronization among all the UAVs such that every member of the group can make autonomous decisions based on updated state information. We posit that such dynamic planning is imperative for mission-oriented platoons. Thus, we have designed our algorithmic framework to be robust to message loss and agent failure such that it could be applied to any mission that requires a UAV group to make decisions based on each other's data.

A. Overcoming Message Loss

Message loss is a practical consideration in real world scenarios and depending on relevant parameters can be a significant issue. As seen in Fig. 1, lack of updates will cause the platoon to stall and outdated information may cause some UAV to make a decision which is not in accordance with the rest of the platoon, causing the entire group to lose synchronization. Thus it is critical that the UAV group has a mechanism that allows each agent to not only detect possible message loss, but also take corrective steps so that the platoon can move forward.

To do this, we have added a timer based *Retransmission* mechanism to each UAV. Each message now consists of a *Sequence Number* such that the UAV can keep track of an expected update. In case of a missed update, the *Retransmission Timer* handles retransmission of messages when required. In our Plume Wrapping example, we focus on the *Stop-Click Phase* which requires synchronization between the agents so that each UAV receives updated state information from every other UAV in order to make accurate decisions. States are monotonically increasing and are tracked using message *Sequence Numbers*. The *Stop-Click Phase* begins with messages numbered 0, which are incremented in each round. If a message with sequence number $\mathcal{P}$ from UAV$_i$ to UAV$_j$ is lost, UAV$_j$ will be unable to progress until it receives $\mathcal{P}^{th}$ state update from UAV$_i$. Progressing to round $(\mathcal{P}+1)$ refers to the UAV calculating its next position based on its own observation and $N-1$ number of $\mathcal{P}^{th}$ density updates. During this time, if UAV$_i$ advances to $(\mathcal{P}+1)^{th}$ round, UAV$_i$ will then broadcast a message with sequence number $(\mathcal{P}+1)$ which will be received by all other UAVs, including UAV$_j$. UAV$_j$ will then use the information from round $\mathcal{P}$ (also stored in the message) to eventually advance to $(\mathcal{P}+1)$. While our algorithm allows UAVs to choose between *Expansion* and *Rotation*, generally speaking, the choice is dependent on the mission being completed. The detailed algorithm can be found in Algorithm 1. Method *processMsg* is triggered when the UAV receives a broadcast message while method *retransmissionTimeOutHandler* is triggered when a timeout occurs, i.e., the UAV has not advanced to the next stage for a predefined time.

Algorithm 1 : Mitigating Message Loss

```
Procedure: processMsg (msg, currentSeq)
 1: $senderSeq \leftarrow getSequence(msg)$
 2: if $senderSeq = currentSeq$ then
 3:     Parse and Store Current Info from $msg$
 4: if $senderSeq > currentSeq$ then
 5:     Parse and Store Previous Info from $msg$
 6: if Received $n$ messages with $currentSeq$ then
 7:     $currentSeq \leftarrow currentSeq + 1$
 8:     Overwrite Broadcast Message $M$ and Send
 9:     Restart Retransmission Timer
10:     Move to Next Target Position
- - - - - - - - - - - - - - - - - - - - - - - - - -
Procedure: retransmissionTimeOutHandler()
 1: Send Current Broadcast Message $M$
 2: Restart Retransmission Timer
```

If further messages are lost, UAV$_i$ will keep retransmitting its own information. UAV$_i$ itself will not be able to move to round $(\mathcal{P}+2)$ until UAV$_j$ advances to $(\mathcal{P}+1)$. Here we note that the reason UAV$_j$ is unable to advance to $(\mathcal{P}+1)$ is that it has yet to receive $\mathcal{P}$ information from some other UAV$_k$. Since UAV$_i$ has advanced to $(\mathcal{P}+1)$, we know that UAV$_k$ must be in either state $\mathcal{P}$ or $(\mathcal{P}+1)$, however UAV$_j$ only has $(\mathcal{P}-1)^{th}$ state information for UAV$_k$ and thus, cannot move forward. This brings us to conclude that, in the combined memory of all the UAVs, there are at most three sequence numbers possible $(\mathcal{P}-1)$, $\mathcal{P}$ and $(\mathcal{P}+1)$. This restriction on advancing ensures that the UAVs will eventually reestablish synchronization despite allowing for some freedom to move forward. A detailed Petri Net model of this framework is shown in Fig. 2.

Each UAV will, thus, make decisions based only on the latest updates from every other UAV, but continue to wait for others to catch up, guaranteeing partial synchronization with bounded number of states. In fact, we have seen that

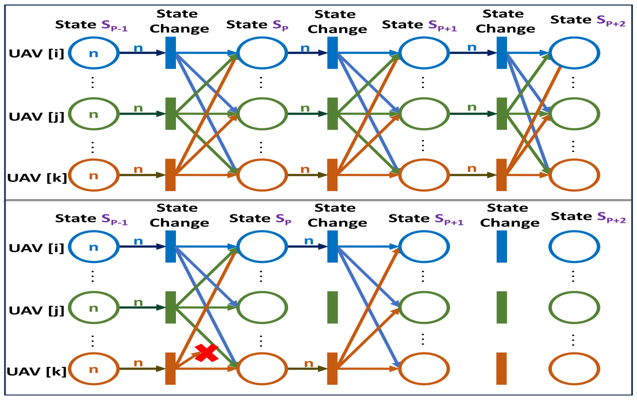

Fig. 2. Petri Net Model of the states of system. The top image shows the ideal case without message loss. Each UAV starts with n tokens representing state information of every group member which are then sent to the State Transition function. Tokens are then distributed to each UAV, representing state information distribution. As a result, each UAV receives n tokens and this process continues. The bottom image shows the case where a message from UAV$_k$ to UAV$_j$ is lost. UAV$_j$ is, thus, unable to advance to State S_{P+1} since it has received information about only State S_{P-1} from UAV$_k$. The other UAVs continue to advance to State S_{P+1} but cannot advance further till they receive updated information from UAV$_j$ thus restricting the number of states possible throughout the collective memory of all UAVs at any given moment to 3, viz., $S_{(P-1)}, S_P, S_{(P+1)}$.

even for a message loss rate of 99%, the UAV platoon re-establishes synchronization eventually. An example of how *Retransmission* mitigates message loss can be seen in Fig. 3.

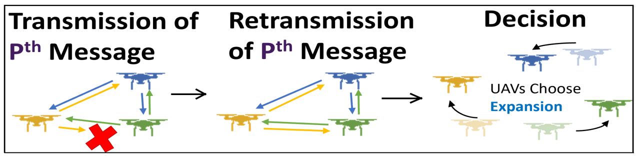
Transmission of Pth Message
Retransmission of Pth Message
Decision
UAVs Choose Expansion

Fig. 3. Example of how UAVs can be resynchronized after Message Loss for the Plume Wrapping Example. With Retransmission, the UAVs do not make a decision till an updated value is received.

B. Overcoming UAV Failure

It is completely possible that, during mission operations, one or more UAV(s) suffer from a component failure. We define failure as a malfunction that necessitates the UAV to leave the platoon and make a safe emergency landing. The emergency mechanism pursued as a result of such failure is out of the scope of our current work.

In order to handle UAV loss, each UAV maintains an *N-bit UAV Loss Timer* for every UAV in the group. The timer for a UAV$_i$ is reset when the current UAV receives a message from it. If a message is not received from some UAV$_i$ by UAV$_j$ for a predefined time, UAV$_j$ marks UAV$_i$ as *lost*. Once a UAV has been marked as lost, the functional UAV removes its information from memory and then continues with the previously described algorithm. To prevent the functional UAVs from accepting outdated updates from UAVs who have not yet detected this loss, we have introduced a *Tamp Down Timer*. This timer is started once a UAV marks another UAV as *lost*. During this time, the functional UAV does not broadcast or process any message. The goal of the *Tamp Down Timer* is to allow every other UAV the time to mark the failing UAV$_i$ as lost, so that further updates are synchronized. We have assumed that the channel is not malicious and does not result in a specific pattern of message losses such that some UAV$_i$ assumes another UAV$_j$ is lost, whereas, UAV$_j$ is functioning. However, even in such a case, UAV$_j$ will assume it has malfunctioning transmitter and safely exit the mission, thus allowing the group to maintain synchronization.

Once the *Tamp Down Timer* expires, each UAV will calculate its next target position based on its current information and $N - M - 1$ positional updates where M refers to the number of UAVs lost. Here, it is possible that some UAV$_k$ advanced to state $(\mathcal{P} + 1)$ based on a failing UAV$_j$'s $\mathcal{P}^{th}$ outdated information. Since other UAVs may not have received this message, we propose that UAV$_k$ *Clicks Back* one step after *Tamp Down Timer* expires. To do this, each UAV checks if the highest sequence number it has received is equal to its own. If yes, this indicates that the UAV may have advanced using data which is no longer relevant. Thus, the UAV needs to recalculate its $(\mathcal{P} + 1)$ position after discarding updates from the lost UAV(s). An example can be seen in Fig. 4. If any UAV stops receiving positional updates, it concludes that its own receiver is broken and safely exits the platoon. As long as at least two UAVs remain functional, the algorithm will reach completion, though the solution may not be practical.

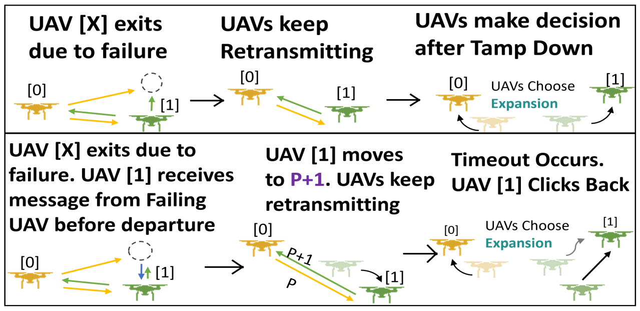
UAV [X] exits due to failure
UAVs keep Retransmitting
UAVs make decision after Tamp Down
[0]
[1]
UAVs Choose Expansion
UAV [X] exits due to failure. UAV [1] receives message from Failing UAV before departure
UAV [1] moves to P+1. UAVs keep retransmitting
Timeout Occurs. UAV [1] Clicks Back
P+1
P

Fig. 4. Example of how UAVs can be resynchronized after UAV Loss for the Plume Wrapping Problem. In the simple scenario (Top), both UAVs wait for a message till timeout occurs and then make their decision. In *Click Back* scenario (Bottom) UAV$_1$ recalculates $(\mathcal{P} + 1)^{th}$ position and moves there.

Alternate Agile Approach

The *Tamp Down Timer* is restarted for every UAV failure which leads to a considerably long waiting period. Over multiple rounds of failure, this problem is exacerbated. To prevent this, we propose an agile algorithm which does not need to wait for a timeout. Instead, when UAV$_i$ detects that UAV$_j$ has malfunctioned, UAV$_i$ records this information in a list and only accepts position updates from other UAVs which have also detected the failure. This is done by including the ID of the failing UAV in subsequent broadcast messages.

V. Simulation & Results

We have designed a multi-agent discrete event simulation in Java of a group of UAVs executing our algorithm while trying to surround a plume. We have varied the number of UAVs used, the percentage of messages lost and the number of UAVs which malfunction when trying to verify the efficacy of our algorithm. Our simulation was executed on a Ubuntu 20.04.1 LTS platform and, for each scenario, we have completed 30 runs and plotted our data with a 95% confidence interval

A. Message Loss

For message loss we varied the chance of a message being lost from 0% to 50% and plotted the time taken to complete the algorithm. With increase in loss of messages, information needs to be retransmitted and thus longer wait time is expected, as seen in Fig. 5. We also notice an increase in time taken as the number of UAVs involved increase.

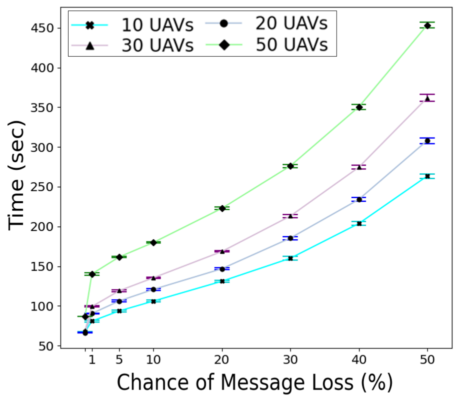

Fig. 5. Time Taken by different UAVs in *Stop-Click Phase* in the presence of message loss.

In Fig. 6, we analysed the time spent by UAVs in each round of *Stop Click Phase*. Overall pattern of waiting remained consistent across different number of UAVs, as well as, different loss circumstances with higher chance of message loss leading to longer wait times. The pattern of sharp spikes we observed were the result of *Rotation* taking more time than *Expansion* due to longer movement by some UAVs. Another expected result was the evidence of more messages being required when there is a higher percentage of message loss. As can be seen in Fig. 7, messages generated increased as message loss increased. This is because any number of messages being lost in a round will result in $N-1$ retransmission messages.

B. UAV Loss

We have also tested the performance of our algorithm when one or more UAVs malfunction and analysed the effect on coordination among UAVs. We have assumed that the channel while lossy, is not malicious and, as such, will not result in distinct patterns of message loss that lead to uncertain behaviours. The probability of such a pattern occurring naturally is vanishingly small and can be adjusted by changing the timeout period set for detecting UAV loss.

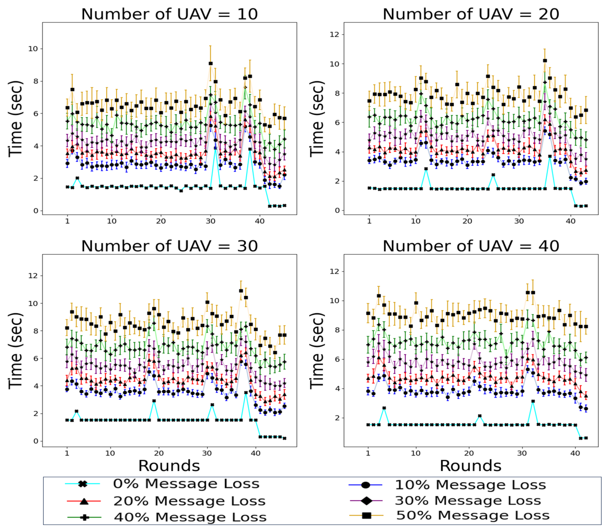

Fig. 6. Time taken per *Stop Click* Round by different UAVs in the presence of varying chances of Message Loss.

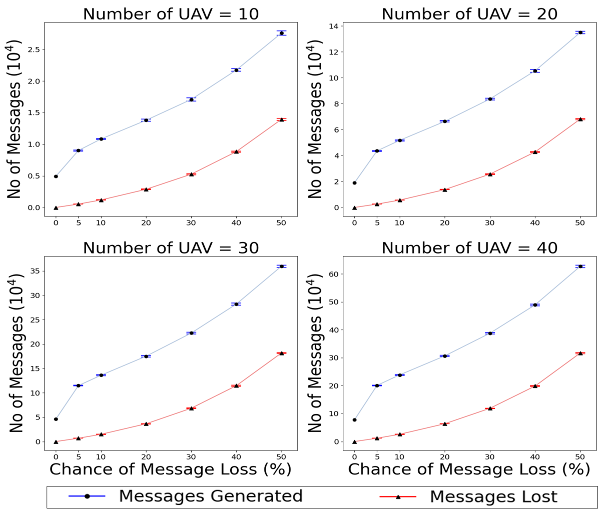

Fig. 7. Total Number of Messages Generated and Lost per UAV

In Fig. 8(a) we have compared the time taken in *Stop Click Phase* for a Single UAV failure over different scenarios of message loss. Our observations show that, the *Tamp Down Mode* takes more time to complete than the proposed *Agile Mode*, and, in both cases a Transmitter fault takes almost the same time as the case where both Transmitter and Receiver begin to malfunction, while only Receiver Fault takes more time. This is because if an UAV does not realize its receiver is faulty, it continues to wait for messages from other UAVs till its own timeout occurs. During this time, the UAV will continue sending messages, thereby, stalling the algorithm for one timeout period.

In Fig. 8(b) we have compared the time taken when five UAVs malfunction to that when zero or one UAV fails. We can see that while failure of more UAVs takes more time, our

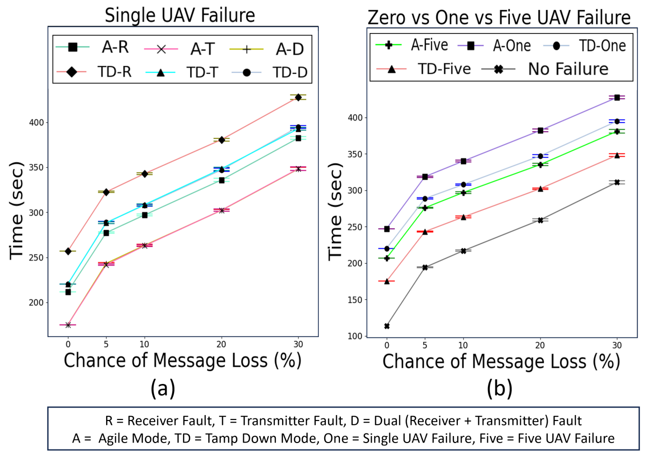

Fig. 8. (a) Time Taken for Single UAV Failure and (b) Comparison between different number of UAV failures.

Agile Mode still outperforms *Tamp Down Mode* in every case. For certain cases of high message loss, we observed that for a specific pattern of loss from the same the UAV, the group would lose synchronization resulting in some members making incorrect choices. However, as stated previously, we expect our channel to be free of such malicious patterns, that result in UAVs incorrectly assuming an agent failed when it was still functioning. For the purposes of our testing, we have limited channel loss chance to 30%, which guarantees correctness.

VI. Conclusion & Future Work

We have designed an algorithm that guarantees partial synchronization in a distributed multiparty system, such as a UAV platoon, in the face of random message loss, and even complete failure of one or more UAVs. Our algorithm is generic and can be applied to platoons engaged in any mission that requires state coordination via sharing of information, as required for autonomous trajectory coordination by the various UAVs in the platoon. Through an exemplar problem, we have shown that our algorithm guarantees mission completion and correctness, with each agent advancing through monotonically increasingly numbered states, and the variation in states among all agents in the platoon being bounded by 3. As message loss probability increases, so does the probability of incorrect re-synchronization after a UAV loss; this can be mitigated by correctly setting timeout duration.

Currently we have considered that every UAV is in communication range of every other UAV in the platoon. We believe it is possible to adapt our approach to the case of limited UAV-to-UAV communication range and multi-hop communications. It would also be of interest to demonstrate an approach for allowing UAV failures to be temporary, such that a failed UAV may rejoin the mission once it has recovered from communications fault. These are all part of our ongoing work. Our generic approach of synchronizing state among the UAVs of a platoon can aid mission algorithm development for a broad range of platoon missions.

References

[1] Y. Du, "Multi-UAV search and rescue with enhanced a algorithm path planning in 3d environment," *International Journal of Aerospace Engineering*, vol. 2023, p. 8614117, 2023. Publisher: Hindawi.

[2] C. Qu, J. Boubin, D. Gafurov, J. Zhou, N. Aloysius, H. Nguyen, and P. Calyam, "UAV swarms in smart agriculture: Experiences and opportunities," in *2022 IEEE 18th International Conference on e-Science (e-Science)*, pp. 148–158, 2022.

[3] X. Zhou, X. Wen, Z. Wang, Y. Gao, H. Li, Q. Wang, T. Yang, H. Lu, Y. Cao, C. Xu, and F. Gao, "Swarm of micro flying robots in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm5954, 2022. Publisher: American Association for the Advancement of Science.

[4] A. Arputha Babu John and R. Dutta, "Cooperative trajectory planning in an intercommunicating group of UAVs for convex plume wrapping," in *2017 IEEE 38th Sarnoff Symposium*, pp. 1–6, 2017.

[5] S. Abdollahzadeh, P.-L. Proulx, M. S. Allili, and J.-F. Lapointe, "Safe landing zones detection for UAVs using deep regression," in *2022 19th Conference on Robots and Vision (CRV)*, pp. 213–218, 2022.

[6] FEMA, "Chemical fire in Apex, North Carolina," USFA-TR 163, US Fire Administration, Apr. 2008. Homeland Security Digital Library.

[7] P. K. Chitumalla, D. Harris, B. Thuraisingham, and L. Khan, "Emergency Response Applications: Dynamic Plume Modeling and Real-Time Routing," *IEEE Internet Computing*, vol. 12, pp. 38–44, Jan. 2008. Conference Name: IEEE Internet Computing.

[8] B. Chatterjee and R. Dutta, "Studying the effect of network latency on an adaptive coordinated path planning algorithm for UAV platoons," in *Proceedings of the Eighth Workshop on Micro Aerial Vehicle Networks, Systems, and Applications*, DroNet '22, pp. 7–12, Association for Computing Machinery, 2022.

[9] J. Capitan, L. Merino, and A. Ollero, "Cooperative decision-making under uncertainties for multi-target surveillance with multiples UAVs," *Journal of Intelligent & Robotic Systems*, vol. 84, no. 1, pp. 371–386, 2016.

[10] Y. Zhao, X. Wang, C. Wang, Y. Cong, and L. Shen, "Systemic design of distributed multi-UAV cooperative decision-making for multi-target tracking," *Autonomous Agents and Multi-Agent Systems*, vol. 33, pp. 132–158, Mar. 2019.

[11] D. Doering, A. Benenmann, R. Lerm, E. P. de Freitas, I. Muller, J. M. Winter, and C. E. Pereira, "Design and Optimization of a Heterogeneous Platform for multiple UAV use in Precision Agriculture Applications," *IFAC Proceedings Volumes*, vol. 47, pp. 12272–12277, Jan. 2014.

[12] Y. Liu, H. Liu, Y. Tian, and C. Sun, "Reinforcement learning based two-level control framework of UAV swarm for cooperative persistent surveillance in an unknown urban area," *Aerospace Science and Technology*, vol. 98, p. 105671, Mar. 2020.

[13] M. Ángel, A. Al-Kaff, P. Flores, D. Martín, and A. de la Escalera, "Software Architecture for Autonomous and Coordinated Navigation of UAV Swarms in Forest and Urban Firefighting," *Applied Sciences*, vol. 11, p. 1258, Jan. 2021. Number: 3 Publisher: Multidisciplinary Digital Publishing Institute.

[14] Z. Fu, Y. Chen, Y. Ding, and D. He, "Pollution Source Localization Based on Multi-UAV Cooperative Communication," *IEEE Access*, vol. 7, pp. 29304–29312, 2019. Conference Name: IEEE Access.

[15] D. Zarzhitsky, D. Spears, and W. Spears, "Swarms for chemical plume tracing," in *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pp. 249–256, June 2005.

[16] S. Huang, R. S. H. Teo, J. L. P. Kwan, W. Liu, and S. M. Dymkou, "Distributed UAV loss detection and auto-replacement protocol with guaranteed properties," *Journal of Intelligent & Robotic Systems*, vol. 93, no. 1, pp. 303–316, 2019.

[17] A. Aminzadeh and A. Khoshnood, "A novel distributed fault-tolerant cooperative control approach for auto-reconfiguration of a multi agent system in natural disasters," in *2021 9th RSI International Conference on Robotics and Mechatronics (ICRoM)*, pp. 163–170, 2021. ISSN: 2572-6889.

[18] S. Huang, W. Cui, J. Cao, and R. S. H. Teo, "Self-organizing formation control of multiple unmanned aerial vehicles," in *IECON 2019 - 45th Annual Conference of the IEEE Industrial Electronics Society*, vol. 1, pp. 5287–5291, 2019. ISSN: 2577-1647.

[19] N. Arnold, J. Gruber, and J. Heitz, "Spherical expansion of the vapor plume into ambient gas: an analytical model," *Applied Physics A*, vol. 69, pp. S87–S93, Dec. 1999.