COMMUNICATION ISSUES IN THE COOPERATIVE CONTROL OF UNMANNED AERIAL VEHICLES

Jason W. Mitchell and Andrew G. Sparks
Flight Control Division
Air Force Research Laboratory (AFRL/VACA)
Wright-Patterson AFB, OH 45433-7531
{Jason.Mitchell,Andrew.Sparks}@wpafb.af.mil

Abstract

Communication needs are considered for the cooperative control of unmanned aerial vehicles with resource allocation performed by a network flow which may be iterative, and may operate synchronously or asynchronously. We briefly outline single and iterative network flow assignment algorithms and their communication requirements. Then, we consider the abstracted communication framework recently incorporated into the MultiUAV simulation package to manage the communication requirements of the allocation algorithms and implementation scenarios. As an example, a model is constructed to investigate the effect of fixed communication delays on the performance of an iterative network flow implemented as a redundant, centralized optimization.

1 Introduction

Coordination and cooperation between unmanned aerial vehicles (UAV) has the potential to significantly improve their effectiveness in many situations. For the typical tasks that a UAV must perform, i.e. detection, classification, attack, and verification, explicit vehicle cooperation may be required to meet specific objectives. Thus, the ability to communicate information between vehicles becomes mission essential and provides an opportunity to enhance overall capability.

While vehicle communications may provide the opportunity to enhance performance, such communication may reduce performance considerably. This is because control algorithms generally are designed without regard to their concomitant communication requirements. For the control system designer, such treatment is undertaken to reduce algorithmic complexity and obtain a manageable result. Consequently, it becomes necessary to quantify the effect of communication on the control algorithms ex post facto. As an example of this design strategy, consider several methods that have been previously studied to produce optimal single task assignments [1, 2], and more recently, the optimal assignment of a sequence of tasks using an iterative network flow model [3]. A common, and often implicit, assumption in these designs is that the vehicle-to-vehicle communication model is perfect, which typically implies that communication is both instantaneous and error free. Unfortunately, any physical realization will almost certainly violate this assumption either due to design criteria or possible adversarial activity.

In this work, communication needs are considered for the cooperative control of unmanned aerial vehicles with resource allocation performed by a network flow, which may be iterative, and may operate synchronously or asynchronously. In the following, we briefly outline the single and iterative network flow assignment algorithms and their communication requirements. Then, we consider the abstracted communication framework recently incorporated into the MultiUAV simulation package [4, 5] to manage the communication structures of the allocation algorithms and implementation scenarios. As an example, a model is constructed to investigate the effect of fixed communication delays on the performance of an iterative network flow implemented as a redundant, centralized optimization.

2 Background

We begin with a short description of a typical MultiUAV simulation scenario and a brief outline of the network flow task allocation models.

The current configuration of MultiUAV simulates, but is not limited to, autonomous wide area search munitions (WASM), which are small UAVs powered by a turbojet engine, with sufficient fuel to fly for a short period of time. They are deployed in groups from larger aircraft flying at higher altitudes. Individually, they are capable of searching for, recognizing, attacking, and verifying targets.

2.1 Scenario

We begin with a set of N vehicles, deployed simultaneously, each with a life span of approximately thirty (30) minutes, and indexed by $i \in \mathbb{Z}[1, N]$. Targets that might be found by searching fall into known classes according to the value or score associated with destroying them. We index them with j as they are found, so that $j \in \mathbb{Z}[1, M]$ and V_j is the value of target j. We assume that there is no precise a priori information available about the number of targets or their locations. This information can only be obtained by the vehicles searching for and finding potential targets via Automatic Target Recognition (ATR) methodologies. The ATR process is modeled using a system that provides a probability that the target has been correctly classified. The probability of a successful classification is based on the viewing angle of the vehicle relative to the target, Rasmussen et al. [5]. For this exercise, the possibility of incorrect identification is not modeled, however targets are not attacked unless a 90% probability of correct identification is achieved. Further details of the ATR methodology can be found in Chandler and Pachter [6], with a detailed discussion available in Chandler and Pachter [7]. Once successfully classified as a target, the attack vehicle is selected. Upon reaching the target to be serviced, the vehicle releases its munition and is subsequently declared an unavailable asset; attack is a terminal task for WASM. The serviced target must finally be verified as destroyed, completing the target specific task chain.

Throughout the simulation, at each target state change or task failure, a resource allocation algorithm is executed to compute task assignments. While the computed assignment is suboptimal, it is hoped that it is on average near-optimal, Rasmussen et al. [8].

2.2 Task Allocation: Network Optimization Model

To model weapon system allocation, we treat the individual vehicles as discrete supplies of single units, tasks being carried out as flows on arcs through the network, and ultimate disposition of the vehicles as demands. Thus, the flows are zero (0) or one (1). We assume that each vehicle operates independently, and makes decisions when new information is received. These decisions are determined by the solution of the network optimization model. The receipt of new target information triggers the formulation and solving of a fresh optimization problem that reflects current conditions, thus achieving feedback action. At any point in time, the database on-board each vehicle contains a *target* set, consisting of indices, types and locations for targets that have been classified above the probability threshold. There is also a *speculative* set, consisting of indices, types and locations for potential targets that have been detected, but are classified below the probability threshold and thus require an additional look before striking.

The model, seen in Figure 1, is demand driven, with the large rectangular node on the right exerting a demand-pull of N units, so that each of the nodes on the left must flow through the network. In the middle layer, the top M nodes represent all of the successfully classified targets, and thus are ready to be attacked. An arc exists from a specific vehicle node to a target node if and only if it is a feasible vehicle/target pair. At a minimum, the feasibility requirement would mean that there is enough fuel remaining to strike the target if so tasked. Other feasibility conditions could also be considered, e.g. heterogeneous weapons or sensing platforms, poor look-angles. The center R nodes of the middle layer represent potential targets that have been detected, but do not meet the minimum classification probability. We call them *speculatives.* The minimum feasibility requirement to connect a vehicle/speculative pair is sufficient fuel for the vehicle to deploy its sensor and sufficiently elevate the classification probability. The lower-tier G nodes model alternatives for verification of targets that have been struck. Finally, each node in the vehicle set on the left has a direct arc to the far right node labeled sink, modeling the option of continuing to search. The capacities on the arcs from the target and speculative sets are fixed at one (1). Due to the integrality property, the flow values are constrained to be either zero (0) or one (1). Each unit of flow along an arc has a benefit which is an expected future value. The optimal solution maximizes total value. For a more detailed discussion, that includes the issue of the *benefit calculation*, see Schumacher et al. [3].

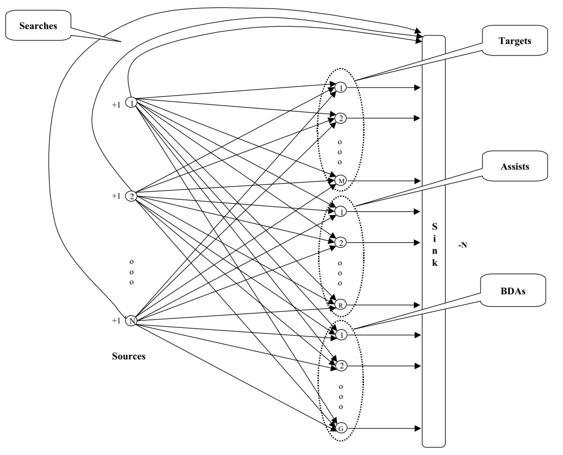

Figure 1: Network flow allocation model.

2.2.1 Single Pass Network Flow

Single task assignment in MultiUAV is formulated as the capacitated transshipment problem (CTP) [1]. Due to the special structure of the problem, there will always be an optimal solution that is all integer [2]. Thus, solutions to this problem pose a small computational burden, making it feasible for implementation on the processors likely to be available on disposable wide area search munitions.

2.2.2 Iterative Network Flow

Due to the integrality property, it is not normally possible to simultaneously assign multiple vehicles to a single target, or multiple targets to a single vehicle. However, using the network assignment iteratively, *tours* of multiple assignments can be determined [3]. This is done by solving the initial assignment problem once, and only finalizing the assignment with the shortest estimated arrival time. The assignment problem can then be updated assuming that assignment is performed, updating target and vehicle states, and running the assignment again. This iteration can be repeated until all of the vehicles have been assigned terminal attack tasks, or until all of the target assignments have been fully distributed. The target assignments are complete when classification, attack, and verification tasks have been assigned for all known targets. Assignments must be recomputed if a new target is found or a munition fails to complete an assigned task.

2.3 Information Requirements

The implementation of the task allocation algorithms outlined above requires communication of information between vehicles. Previous versions of MultiUAV used to investigate optimal task allocation assumed perfect and globally accessible information about vehicle and target states. From many perspectives, perfect information and instantaneous access is unrealistic, particularly when considering physical communication and processing constraints. Consequently, to determine the effects of physically realizable systems, it is necessary to understand what information is necessary for the algorithms to function.

Since both algorithms discussed here make use of network flow, the necessary information is common between them. The overarching optimization problem can be characterized as centralized and redundant, i.e. each vehicle computes its own network flow. Momentarily disregarding communication issues, in general, the problem requires a synchronized database of target and vehicle state information. With this, each vehicle computes the benefits for the arcs in the network, and solves the optimization problem to maximize the total benefit. From the previous MultiUAV implementation, the information that must be communicated between vehicles is: ATR data; target and vehicle positions; target, vehicle, and task status; and vehicle trajectory waypoints.

Having identified the necessary information, we can begin to consider the effect of communicating that information between vehicles. We recall that each vehicle much have a synchronized information database to ensure that the task assignments resulting from the network flow are consistent. Clearly, if a vehicle perceives its task space differently, it is unlikely that it will produce the same task assignment as the synchronized vehicles. This raises a host of important questions, e.g. Can team members successfully cooperate in the presence of delayed information? How much delay can be tolerated? Is there any benefit to limited asynchronous operation? Continuing with this line of inquiry, we begin to ask very fundamental questions: How do we define and measure cooperation? How can vehicles decide who is cooperative or non-cooperative? These questions are difficult to answer and are subject to many, sometimes competing, constraints. Initially, we would like to quantify the effect of delayed information on the performance of the existing, and well tested, cooperative control algorithms available. To do this, we turn to the MultiUAV simulation package.

3 Simulation Framework

The MultiUAV [5] simulation package is capable of simulating multiple unmanned aerospace vehicles which cooperate to accomplish a predefined mission. The purpose of the package is to provide a simulation environment that researchers can use to implement and analyze cooperative control algorithms. The simulation is composed as a hierarchical decomposition where inter-vehicle communication is explicitly modeled. The package includes plotting tools and links to an external programs for post-processing analysis. Each of the vehicle simulations include six-degree-of-freedom dynamics and embedded flight software (EFS). The EFS consists of a collection of *managers* or *agents* that control situational awareness and responses of the vehicles. In addition, the vehicle model includes an autopilot that provides a waypoint navigation capability. In its original form, MultiUAV [4] could simulate a maximum of eight (8) vehicles and ten (10) targets, however recent work eases the previous burden of extending these limits. The EFS managers implement the cooperative control algorithms, including the iteratively applied CTP algorithm previously discussed. The individual managers contained within the vehicles include: Tactical Maneuvering, Sensor, Target, Cooperation, Route and Weapons.

The simulation was constructed using SIMULINK and MATLAB[1]

[1]see The Mathworks website at `http://www.mathworks.com/`.

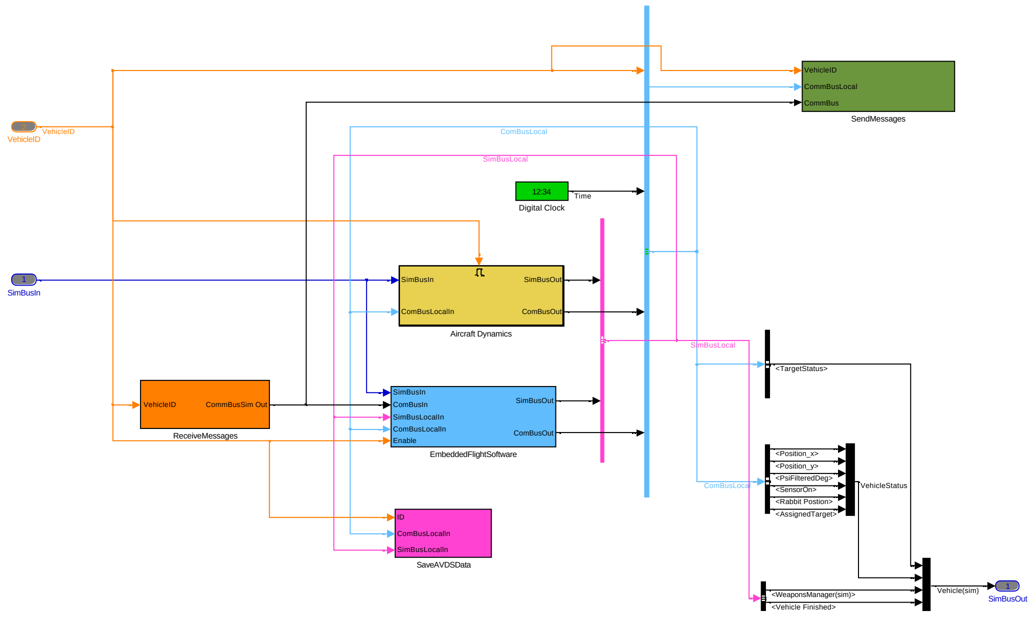

Figure 2: Vehicle model with communication simulation ability.

3.1 Modeling Vehicle Communications

Maximum design flexibility is a significant and yet vague requirement that must be met by any potential communication abstraction. By maintaining genericity, we ensure that the resulting solution will accommodate the simulation of specific communication requirements, e.g. protocol-specific, theater-specific, or hardware-specific, while providing a simple and general framework to quantify vehicle-to-vehicle communication needs, e.g. peak or average data-rate.

Unlike the vehicle model seen in Figure 2, previous releases of the simulation provided vehicle-to-vehicle communication via a signal bus denoted by `CommBus`, while a second aggregated signal bus, labeled `SimBus`, contained the *truth* information for the simulation. The combination of these two data buses represented the complete information state of the simulation. This *perfect* information state was available to all vehicles at every simulation time-step. As previously mentioned, this global and instantaneous information access is unrealistic. As part of recent improvements to MultiUAV, a new communication framework is introduced for remote communication [5], which can be seen in Figures 2–4.

3.2 Design Abstraction

To provide flexibility in implementation of communication simulations that contain varying levels of detail, a generic message passing scheme was chosen as the Virtual Communication Representation (VCR). In this design, specific message types and their format are defined centrally in the VCR and made globally available to the various managers as context requires[2]. Minimally, a message definition must contain a unique message identifier, time-stamp(s), message layout enumeration, and data field to be written by the manager context. Particular messages may be selected by the manager context as output resulting from a computation that must be remotely communicated. Outgoing messages [Fig. 3], which include data, from each vehicle are stored centrally, and pointers to these messages are distributed to an individual input queue for each vehicle. These pointers are composed of the original message header and

[2]The message structure discussed here refers to the format dictated by the MultiUAV package, rather than to messages related to a specific communication system model.

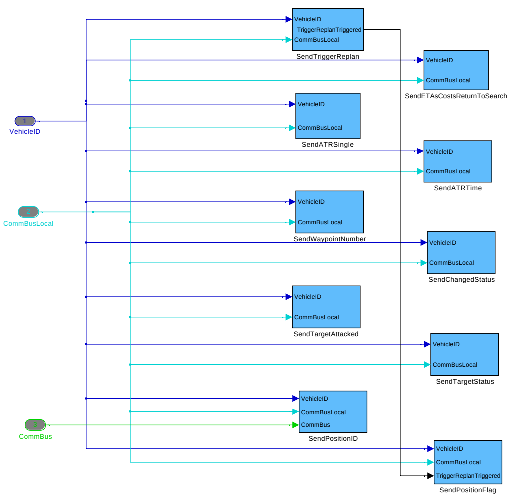

Figure 3: SIMULINK `SendMessages` block.

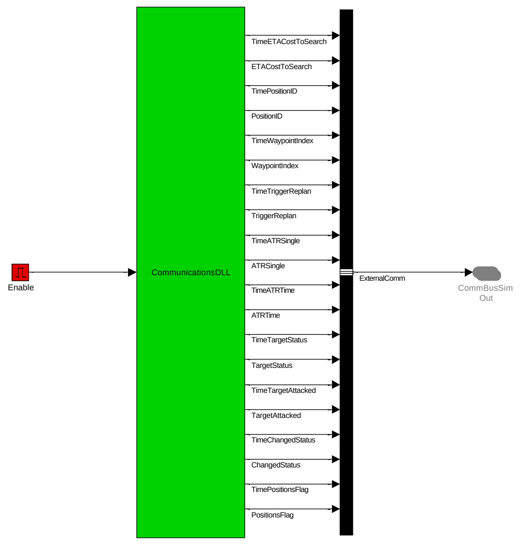

Figure 4: SIMULINK `ReceiveMessages` block.

should minimally inform the receiver of the message type, time sent, quality or priority of the message, and which central repository contains the associated message data. A user defined rule component controls the distribution of incoming messages to all vehicles based on the message headers, see Figure 4.

We avoid adhering to a specific communication model in MultiUAV by isolating the message delivery rules in user controlled components. Thus, end-users are free to choose any preferred communication model. Moreover, the genericity of the VCR specification provides for easy extension.

4 Simulation

To compare the performance of the three-deep iterative CTP assignment algorithm with and without communication delays, we chose four particular cases of fixed delays and applied a MonteCarlo approach consisting of fifty (50) simulations each [8, 9]. These simulations, for same initial seed, are compared using attack and verification task completion as metrics. Additionally, we directly compare two (2) same seed simulations for each of the delay cases.

4.1 Cases

Case	Comm Delay
0	0 s
1	1 s
2	2 s
3[a]	2 s

[a]Includes ≈ 0.5 s processing delay on originating vehicle.

Table 1: Delay cases.

The MultiUAV scenario chosen was three (3) vehicles and two (2) targets. The targets are placed randomly in a search-box, while the vehicles are aligned and tasked to search in a *lawn-mowing* fashion for a maximum mission time of 200 s. This scenario was then run for each of the communication delay cases shown in Table 1.

The specific delay values above were chosen largely out of convenience, however they are intended to represent a significant delay as compared to the lifetime of the vehicle in question. In this way, we hope to clearly quantify the effect such delays have on the cooperative control algorithms.

The self-processing delay of case 3 indicates that when a vehicle sends new information to remote vehicles, it may require additional processing time before the vehicle arrives at a

Case (#)	None (%)	1-K (%)	2-K (%)
0	0.0	100.0	72.0
1	0.0	100.0	6.0
2	0.0	100.0	14.0
3	4.0	96.0	36.0

Table 2: Percentage of successful attack

Case (#)	None (%)	1-V (%)	2-V (%)
0	0.0	100.0	54.0
1	64.0	36.0	4.0
2	48.0	52.0	2.0
3	36.0	64.0	8.0

Table 3: Percentage of successful verify

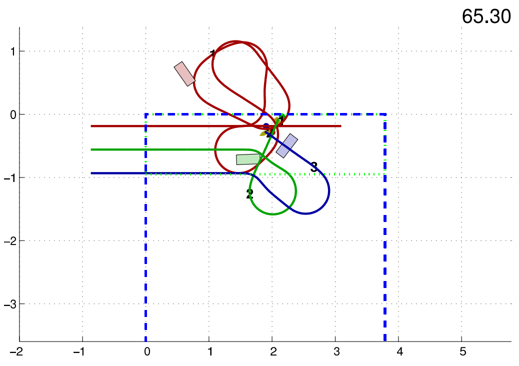

Figure 5: Delay case 0 for seed 1.

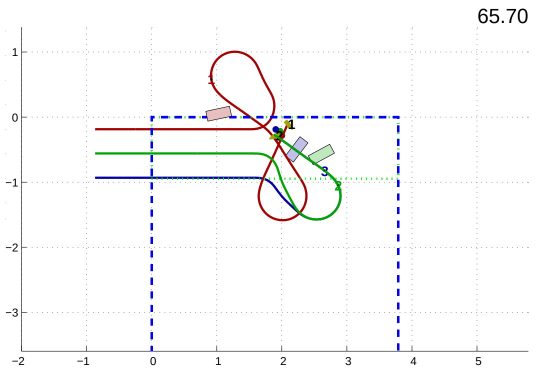

Figure 6: Delay case 1 for seed 1.

decision. This, of course, is an initial step in moving all internal vehicle communication from a signal to a message base.

For all simulation runs, replanning is triggered only by the change of a target state. No timed or other replans are injected into the system.

4.2 Results

The summary data, shown in Tables 2 and 3, provides some interesting results. With no delay, we see that both targets are attacked approximately 72% of the time before exceeding the mission timer. Also, there is always at least one verify, and both targets are verified approximately 54% of the time prior to maximum mission time. For the delay cases, the percentage of both targets successfully being attacked is considerably reduced. This is largely due to a *target sink effect* induced by the delayed information because the vehicles no longer maintain a central information repository, thus the same task may be assigned to multiple vehicles. Since the vehicles keep their current task list until a target state changes, it is possible for the information to arrive too late to be of use, unintentionally resulting in multiple classifies, attacks, and verifies on a target. This becomes a significant problem in areas of high target density and with tight vehicle spacing. Surprisingly, as the delay increases, the percentage of two (2) successful attacks increases. For high target density and close vehicle spacing, the added communication delay results in slightly longer paths to the target, providing a window for new information to arrive before the vehicle acts on its replanned decision. The trend is similar for verification tasks, except that the number of two (2) target verifies is greatly reduced. This was due to a combination of the target sink effect, and incomplete tasks at maximum mission time.

It is more instructive to compare specific simulations individually. Qualitatively, this provides less ambiguous information regarding the effect of the communication delay. For each of the delay cases, simulation results for the first and last seed values were plotted.

For seed 1, we see the CTP selected paths in Figs. 5–8, where the graph scale is in miles. With the introduction of delay 1 [Fig. 6], we see the previously mentioned sink behavior as vehicles two (`V2`) and three (`V3`) both strike target two (`T2`), while vehicle one (`V1`) has a sufficient travel distance to receive the attack information and replan to verify `T2`, then attack target one (`T1`). For delay 2, seen in Fig. 7, there is little difference from Fig. 6. Finally,

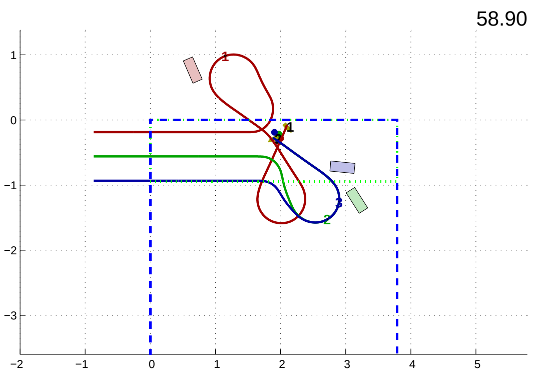
Figure 7: Delay case 2 for seed 1.

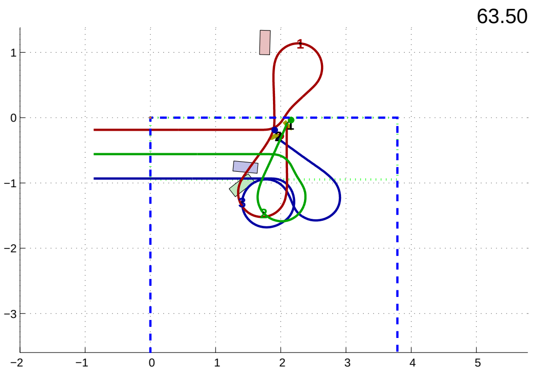
Figure 8: Delay case 3 for seed 1.

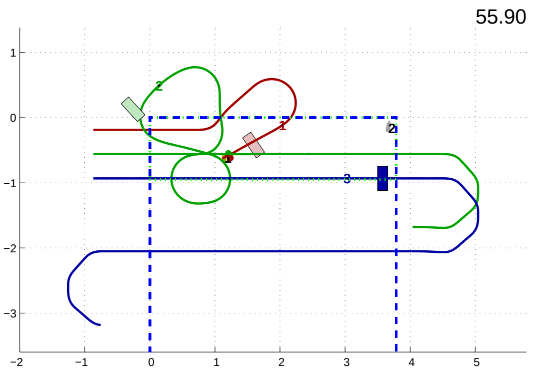
Figure 9: Delay case 0 for seed 50.

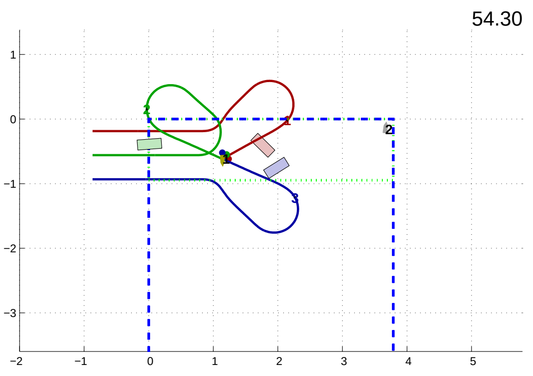
Figure 10: Delay case 1 for seed 50.

with delay 3, seen in Fig. 8, we notice that `V2` attacks `T1`, `V3` attacks `T2`, and `V1` verifies the attack on `T2` and is scheduled to verify the attack on `T1`. Unfortunately, the maximum time is reached prior making the final verification. Interestingly, `V3` displays a *churning* behavior as a result of missing the initial replanned waypoint, i.e. `V3` had passed the waypoint before it was received, resulting in a minimum turn radius circle to reacquire the planned path. This raises the possibility that a vehicle could churn forever if a missed waypoint fell sufficiently far inside the minimum turn radius.

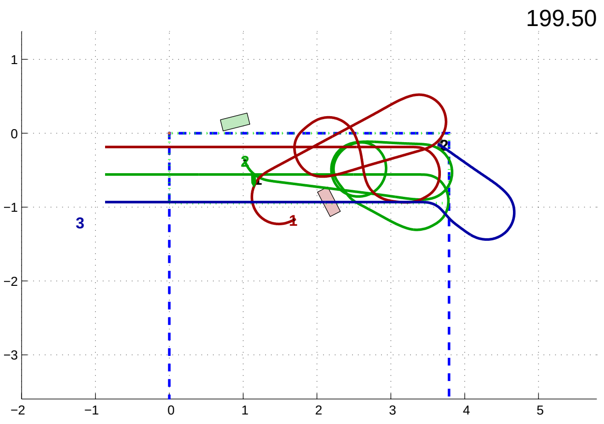
Figure 11: Delay case #3 for seed value #50.

Considering seed 50, the selected paths are seen in Figs. 9–11, again with a graph scale in miles. With no delay, `T1` is successfully prosecuted, however `T2` remains undetected since it resides in the search-lane of `V1`, who attacked `T1`. For delay 1 seen in Fig. 10, we again see the sink effect but for a single target, leaving vehicle spacing as the culprit; delay 2 produced similar results. The paths for delay 3, seen in Fig. 11, are quite convoluted. In this case, the self-delay results in an unintentionally non-communicated message. As `V2` detects `T1`, it does not trigger a position update. Thus, the vehicles find no need to replan. Upon discovering `T2`, a position update is triggered and a successful replan occurs. While `T2` is prosecuted quickly, the verification is much later because `T1` requires multiple passes to classify for attack, i.e. low ATR values. As before, we see a churning loop for V2 resulting from a delayed replan waypoint.

4.3 Improvements

With this first attempt to determine the effect of communication delays on the existing cooperative control algorithms available within the MultiUAV package, several deficiencies became apparent.

The first issue was that of the target sink effect, resulting from the lack of a synchronous environmental perception. A direct way to prevent this behavior would be to require message handshaking. This would provide a opportunity for the vehicles to verify their perception and reach a consensus before individually computing the task assignment list. Unfortunately, this will not insulate the team from the possibility that a given vehicle may difficulty communicating, e.g. adversarial activity or no line-of-sight.

The second issue was the churning motion that resulted from a vehicle receiving new plans too late. When a vehicle receives position information, to perform a replan, the positions are simply extracted from the message and used without regard to the time-stamp indicating send time. Since the vehicle knows both when the message was sent and received, and could request the vehicles' waypoint lists, it could assume the vehicles continue on the trajectory of their most recent assignment, and filter the positions to the current time. Taking this idea slightly further, by computing the assignment for a near future time, the vehicles could also account for local processing time. The resulting plans should generate waypoints that the vehicles can reach without churning. Of course, we could expect this to cause difficulty if a target status update occurred between the time the filtered-replan took place and new messages were checked, or if the vehicles did not have an accurate estimate of both the solution processing time and existing communication delays.

For assigned tasks derived from delayed information, further improvement may be made by transmitting each vehicles task list after a replan. This would provide a handshaking mechanism to verify a successful replan by ensuring consensus, thus disambiguating same-task assignments that could result in sink behavior. However, considerable care would need to be exercised in implementing such a mechanism so as to avoid circular replanning cycles, perhaps requiring that the vehicles could *agree-to-disagree*.

A less obvious option for improving the performance of the vehicles would be to reformulate the operation of the cooperative control algorithms. In the current implementation, the vehicles all perform a redundant, centralized optimization in which each vehicle computes the trajectories and benefits for each vehicle-target-task combination. One approach would seek to distribute the computation of the benefits, so that each vehicle computed only its own costs and communicated those to the remaining vehicles. This could also serve as the handshaking mechanism that initiates the resource allocation computation.

5 Conclusion

In this paper, a communication model, which was incorporated into the MultiUAV simulation package for wide area search munitions, was used to study the effect of fixed-time communication delays on the performance of an iterative network flow optimization model. This network flow model results in a sequence of linear programs for the optimal allocation of consecutive assignments. The resulting delayed system performance was compared to the performance of a system with a perfect communications model for a set of three vehicle, two target scenarios. The effect of the delayed communication was seen as a significant decrease in successful attack and verify task completion. Most notably, we saw a target sink effect that resulted in a task being performed more than once on a target due to the absence of synchronized information. The typical sink behavior resulted in a target receiving two or more classify-attack task combinations resulting from shorter delays and close target or vehicle proximity. This result is not particularly surprising since a lack of information implies a lack of cooperation. Further delay of the information improved the performance slightly by giving the vehicles a larger decision window before acting on a particular task assignment. In addition, vehicle churning was observed as a result of replan waypoints arriving too late. While the observed cooperation showed significantly degraded performance, several improvements to address these issues became apparent, and were discussed.

6 Acknowledgment

This work was performed while the first named author held a National Research Council Research Associateship Award at the Air Force Research Laboratory (AFRL) in the Air Vehicles Directorate Control Theory Optimization Branch (VACA) located at the Wright-Patterson Air Force Base.

References

[1] Schumacher, C., Chandler, P., and Rasmussen, S., "Task Allocation for Wide Area Search Munitions Via Network Flow Optimization," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2001.

[2] Nygard, K., Chandler, P., and Pachter, M., "Dynamic Network Flow Optimization Models for Air Vehicle Resource Allocation," *Proceedings of the Automatic Control Conference*, 2001.

[3] Schumacher, C., Chandler, P., and Rasmussen, S., "Task Allocation for Wide Area Search Munitions Via Iterative Network Flow Optimization," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2002.

[4] Rasmussen, S. and Chandler, P., "MultiUAV: A Multiple UAV Simulation for Investigation of Cooperative Control," *Proceedings of the Winter Simulation Conference*, 2002.

[5] Rasmussen, S., Mitchell, J. W., Schulz, C., Schumacher, C., and Chandler, P. R., "A Multiple UAV Simulation for Researchers," *Proceedings of the AIAA Modeling and Simulation Technologies Conference*, 2003.

[6] Chandler, P. R. and Pachter, M., "UAV Cooperative Classification," *Workshop on Cooperative Control and Optimization*, Klewer Academic Publishers, 2001 (to appear).

[7] Chandler, P. R. and Pachter, M., "Hierarchical Control of Autonomous Teams," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2001.

[8] Rasmussen, S., Chandler, P. R., Mitchell, J. W., Schumacher, C., and Sparks, A. G., "Optimal vs. Heuristic Assignment of Cooperative Autonomous Unmanned Air Vehicles," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2003.

[9] Mitchell, J. W., Schumacher, C., Chandler, P. R., and Rasmussen, S., "Communication Delays in the Cooperative Control of Wide Area Search Munitions via Iterative Network Flow," *Proceedings of the AIAA Guidance, Navigation, and Control Conference*, 2003.