Scheduling for Offloading Safety-Critical Applications Within Networked Groups of Vehicles

John B. Mains, Thanakorn Khamvilai, and Eric Feron

***Abstract*—This paper presents a combined scheduling and path planning scheme for offloading of safety-critical applications in a group of networked vehicles. Due to the dynamic nature of such systems, a scheduler for applications to be offloaded needs to account for the changes of the network parameters. Such a scheduling scheme based on a mixed-integer program (MIP) is detailed with attention to the time-varying duration of message passing operations. Given knowledge of the scheduling requirements of applications to be offloaded, a path planner for a group of networked vehicles can ensure the resulting path will yield a feasible schedule for offloading. An method for generating such a path is described, utilizing the scheduling scheme and a mixed-integer program for path planning with integral communication capacity constraints. The combined scheme is demonstrated in a case study consisting of a pair of UAS equipped with a line-of-sight inter-vehicle a communication link and onboard processors. We show how the scheme can be used to make individual vehicles of the group fault-tolerant to processor failures.**

***Index Terms*—communication-aware path-planning, fault-tolerance, scheduling, offloading**

I. Introduction

In this paper, we consider the problem of scheduling and path-planning for offloading safety-critical applications within networked groups of vehicles. Offloading in this context refers to the processing of a vehicle-specific application using an offboard processor. Offloading safety-critical applications is of particular interest with respect to fault-tolerance. In the event of a computing hardware failure onboard a member vehicle, that vehicle would no longer be functional as its application would not be computed in time, and may even cause a mission failure for the entire group. If instead the application can be offloaded to another processor, the vehicle with a computing hardware failure can continue its mission. This fault-tolerance is dependent upon being able to offload, which is itself dependent upon network communication capacity and the offboard processor.

If the vehicle belongs to a networked group, offloading within that group allows the vehicle to better ensure that the resources required for fault-tolerance through offloading are available. Two specific resources that must be available are sufficient communication channel capacity and processor time to ensure the results are returned by the deadline. The communication channel that carries the messages containing the input and output of the application must maintain the capacity to do so, or the results may reach the vehicle past the deadline. When offloading within a group of vehicles, the main factor affecting the capacity and evolution of the communication channel between vehicles in the physical arrangement of the group. Thus, sufficient channel capacity for offloading can be ensured by incorporating constraints for the channel capacity into the path planning of the group.

Ensuring sufficient channel capacity is not enough, as the offboard processor must be available to process the application in time to return results before the deadline. If the applications for all vehicles within the group are scheduled together, the resulting schedule can guarantee that the processors are available, given complete control otherwise. In order to find a schedule that is feasible, the scheduler can account for the communication of task results between vehicles and processors as message passing tasks. As the capacity of the communication channels between vehicles varies with their positions, and thus over time in a predictable manner, the scheduler needs to be able to account for the time-varying duration of the message-passing tasks. If the scheduler does not account for the time-varying nature of the message-passing tasks, then the resulting schedule may be infeasible, when the scheduling problem is feasible. A scheduling problem is infeasible if a complete scheduling method, such as a mixed-integer program, cannot find a feasible schedule, one in which every task is completed prior to the deadline. To guarantee that any offloading will be successful, the resulting scheduling problem must be feasible, and if it is not, there must be a method to find a path for the vehicles such that the scheduling problem is feasible if such a path exists.

Previous work in this area falls into three categories. First, there has been recent interest in the problem of offloading with vehicles in the path planning and task scheduling domains [1][2]. Second, scheduling for time-varying systems has been examined, but that has been limited to non-deterministic systems [3][4]. Finally, there has been a significant amount of work path planning with communication constraints, with work on capacity, line-of-sight, and probabilistic models [5][6][7]. The main contributions of the work are a scheduling method that accounts for time-varying tasks, a path planning method with integral communication constraints, and a method to combine the two in order to find a path that is feasible for offloading.

The remainder of the paper is organized as follows: Section

The authors would like to thank General Atomics for the gift that supported this work.

John B. Mains is a student at the Georgia Institute of Technology, Thanakorn Khamvilai is a visiting student at King Abdullah University of Science and Technology and a PhD candidate at the Georgia Institute of Technology, and Eric Feron is a Professor of Electrical Engineering at King Abdullah University of Science and Technology. Email addresses: {jmains3, khamvilai}@gatech.edu, eric.feron@kaust.edu.sa

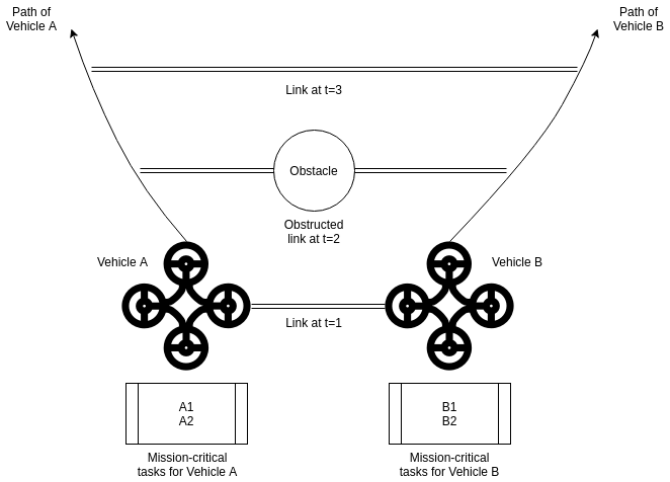


Fig. 1: Two vehicles with paths, an application set, and a time-varying communication link that is blocked by an obstacle.

II covers the scheduling method for time-varying processor networks, Section III covers the path planning method with integral communication constraints, and Section IV presents a case study of a pair of UAS in two situations: 1) when both have an onboard processor and 2) when only one has an onboard processor, indicating a failure.

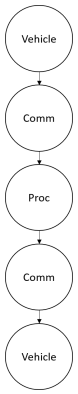
II. Scheduling and Assignment of Applications on Time-Varying Processor Networks

A. Problem formulation

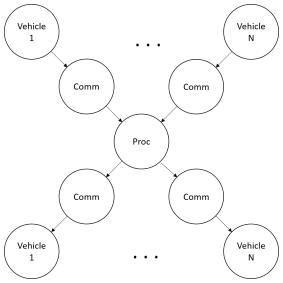
For this application, we consider the problem of finding a schedule and assignment of heterogeneous, precedence related tasks with time-varying duration onto arbitrarily connected machines that minimizes makespan. More formally, given a set of N tasks and M machines and a scheduling window starting at time $t = 0$ and ending at time $t = W$, for each task i find a starting time $s_i \in [0, W]$ and assignment p_i such that the completion time of each task C_i is within the window, $C_i \leq W$ that mimimizes the maskespan and satisfies typing and precedence constraints. Makespan is the time between the starting time of the earliest task to start and the completion time of the latest task to finish, $\max_i C_i - \min_i s_i$. Without release time constrains, the makespan is equivalent to the completion time of the latest task to finish, $\max_i C_i$. The completion time of a task C_i can be represented as $C_i = s_i + d(s_i)$ where $d(s_i)$ is the time-varying duration of the task. The completion time of task i can also be represented by a completion time function $C_i = \delta(s_i)$, which is interchangeable with the time-varying duration representation with $d(t) = \delta(t) - t$. The completion time function representation is beneficial as it allows us to find a convex approximation to be used with the scheduler, as opposed to solving for a schedule with the exact time-varying duration, which may be intractable.

The scheduling problem is heterogeneous, that is a given task can only be executed on a subset of the machines representing its type, in order to account for the heterogeneous nature of a task set composed of processing, message-passing, and ownership tasks. For an application that can be offloaded, we represent the actual processing of an application with a processing task, the sending of the inputs to the assigned processor and the outputs back to the owner with message-passing tasks, and the initiation and conclusion of an application on the owner with ownership tasks. While the processing tasks can be assigned to any processor, and the message-passing tasks assigned to any communication link, the ownership tasks are only related to their specific vehicle, so there are two ownership tasks of a vehicle-unique type per application.

The correct ordering of execution of these tasks is‘ enforced using precedence relations, which describe what other tasks must complete before a given task starts. For an application that can be offloaded, an ownership task for completion must be preceded by a message-passing task for the output, which must be preceded by a processing task for the application itself, which is preceded by a message-passing task for the input, which is preceded by an ownership task for the initiation. This graph of precedence relations for an application tied to a single vehicle can be seen in Figure 2a, while the precedence relations for an application centralized among V vehicles can be seen in Figure 2b. Arbitrarily connected machines, also referred to as machine precedence, refers to restriction on the assignment of a task based upon what the precedence related parent tasks were assigned to and the machine connection graph. For an application that can be offloaded, the message-passing tasks must assigned to machines representing the communication channels that are connected to the ownership assigned machines representing the vehicles and processing assigned machines representing the processors.



(a) Precedence graph for a single vehicle-specific application, whose ownership task type is ”Vehicle”.



(b) Precedence graph for N vehicles, whose ownership task types are ”Vehicle 1” to ”Vehicle N”, that share a centralized task.

Fig. 2: Precedence graphs for different application breakdowns.

B. MIP formulation

In order to solve this problem we use a mixed-integer programming based method with a quadratic program for finding an approximation of the completion time function.

1) Decision variables: The decision variables for this program are the starting time s_i and the machine assignment p_i for each task $i \in N$, with an objective to minimize makespan, and thus the program starts as:

$$\min \max_i \quad C_i \tag{1}$$

$$\text{s.t.} \quad s_i \geq 0 \tag{2}$$

$$C_i \leq W. \tag{3}$$

2) Completion time: A key challenge to scheduling this problem is choosing a form for completion time constraint, specifically a form that encodes the completion time function δ. One way to encode this for a task with time-varying duration is to instead transform the scheduling problem into an assignment problem, using the exact values of the sampled completion times. Encoding the completion time in in this form is intractable for problems with a large number of tasks and samples, as it introduces a large number of integer variables to the program. Alternatively, the completion times can be modeled using a worst case processing time (WCPT) approximation, wherein a constant duration is chosen of the form $d = \max_{t\in[0,W]} \delta(t) - t$. Using the WCPT approximation does not always yield a feasible schedule when one is possible, and can yield schedules with long makespans for tasks with time-varying duration. In order to efficiently and effectively solve problems involving time-varying duration, a better approximation must be found. To this end, we use an affine over-approximation of the form

$$\bar{\delta}(t) = k_1 t + k_2 \geq \delta(t), \tag{4}$$

found by the quadratic program

$$\min_{k_1,k_2\geq 0} \quad \|\boldsymbol{e}\|_2^2 \tag{5}$$

$$\text{s.t.} \quad \boldsymbol{e} = \begin{bmatrix}\boldsymbol{t} & 1\end{bmatrix}\begin{bmatrix}k_1\\k_2\end{bmatrix} - \boldsymbol{\delta} \tag{6}$$

$$\boldsymbol{e} \geq 0 \tag{7}$$

$$k_1 W + k_2 = W + d, \tag{8}$$

where $\boldsymbol{t}$ is vector of sample times of length T (the number of sample times), 1 is a vector of ones of length T, and $\boldsymbol{\delta}$ is a vector of the samples of $\delta(t)$ at the sample times given by $\boldsymbol{t}$. Having found the coefficients k_1 and k_2, the affine over-approximation can be encoded in the program by the constraint

$$C_i = k_1 s_i + k_2. \tag{9}$$

3) No-overlap constraint: When scheduling, the machines are considered fully dedicated, so there can be no-overlap between tasks in time. In order to enforce this, we introduce a binary decision variable, which we wish to define as

$$\sigma_{i,j} = \begin{cases} 1, \; s_j \geq C_i \\ 0, \text{ otherwise.} \end{cases} \tag{10}$$

This definition is enforced in the program using the constraint

$$s_j - C_i - W(\sigma_{i,j} - 1) \geq 0 \quad \forall i \neq j \tag{11}$$

and is combined with the constraint

$$\sigma_{i,j} + \sigma_{j,i} = 1 \quad \forall i \neq j, \tag{12}$$

to enforce the no-overlap constraint. This method is known as a way to handle disjunctive constraint, as introduced by Balas [8]. With the no-overlap decision variable, we can also enforce precedence relations by adding the constraint

$$\sigma_{i,j} \geq a_{i,j} \quad \forall i \neq j, \tag{13}$$

where $a_{i,j}$ is 0-1 parameter of the scheduling problem indicating that task i must precede task j.

4) Heterogeneous machine assignment: In order to find a machine assignment that satisfies typing requirements, we introduce a binary decision variable that we wish to define as

$$x_{i,k} = \begin{cases} 1, \; p_i = k \\ 0, \text{ otherwise} \end{cases} \tag{14}$$

$$\epsilon_{i,j} = \begin{cases} 1, \; p_i < p_j \\ 0, \text{ otherwise.} \end{cases} \tag{15}$$

These definitions are enforced in the program using the constraints

$$p_i = \sum_{k=1}^{M} k x_{i,k} \tag{16}$$

$$p_j - p_i - \epsilon_{i,j} M \leq 0 \quad \forall i \neq j \tag{17}$$

$$p_j - p_i - 1 - (\epsilon_{i,j} - 1)M \geq 0 \quad \forall i \neq j. \tag{18}$$

In order to enforce the no-overlap constraint in a multi-machine setting, the following constraints are added to the program:

$$\epsilon_{i,j} + \epsilon_{j,i} \leq 1 \quad \forall i \neq j \tag{19}$$

$$\epsilon_{i,j} + \epsilon_{j,i} + \sigma_{i,j} + \sigma_{j,i} \geq 1 \quad \forall i \neq j. \tag{20}$$

With the assignment variable defined, we enforce the heterogeneous constraint in the program using

$$x_{i,k} \leq u_{i,k}, \tag{21}$$

where $u_{i,k}$ is a 0-1 parameter of the scheduling problem that represents a match between the type of task i and machine k.

5) Machine precedence: In order to enforce the restrictions of machine precedence, we introduce a final binary decision variable that we wish to define as

$$\gamma_{i,j,h,k} = \begin{cases} 1, \; p_i = h \text{ and } p_j = k \\ 0, \text{ otherwise.} \end{cases} \tag{22}$$

This definition is enforced by the constraints

$$x_{i,h} - \gamma_{i,j,h,k} \geq 0 \quad \forall i \neq j \tag{23}$$

$$x_{j,k} - \gamma_{i,j,h,k} \geq 0 \quad \forall i \neq j \tag{24}$$

$$x_{i,h} + x_{j,k} - 1 - \gamma_{i,j,h,k} \leq 0. \tag{25}$$

This definition is of a similar form to that used by Davidovic to incorporate communication delays when not considering

message-passing as independent tasks [9]. With this definition, we enforce machine precedence by adding the constraint

$$a_{i,j}\gamma_{i,j,h,k} \leq b_{h,k}. \tag{26}$$

The resulting program has a total of $N(N(M^2+2)+M+1)$ variables, of which $N(N(M^2+2)+M)$ are integer variables, and $N(2NM^2+2(N-1)M^2+8N-3)$ constraints. Note that if the approximation had not been used, and instead a time assignment scheme, there would be instead $N(N(M^2+2)+MT)$ variables, all integer, where T is the number of time samples in the scheduling window.

III. Multi-vehicle path planning with integral communication capacity constraints

A. Problem formulation

We consider the problem of finding a path for multiple vehicles modeled as dynamical systems that minimizes control effort while obeying initial state, final state, obstacle avoidance, separation, and integral communication constraints. More formally, given a set of V vehicles and a scheduling horizon t_f, for each vehicle i find a sequence of states $x_i(t) \in \mathbb{R}^n$ and inputs $u_i(t) \in \mathbb{U} \subset \mathbb{R}^m$ for all $t \in [0, t_f]$ while satisfying the constraints. In this scope, we are concerned with discreted paths, in which the time window is discretized by a time step δ_t, such that there are a total of $T = t_f/\delta_t$ time steps in the planning window.

The vehicles are modeled as dynamical systems for path planning, such that

$$x_i(t+1) = f(x_i(t), u_i(t)). \tag{27}$$

Without loss of generality, hereafter we work with discrete time linear time invariant systems of the form

$$x_i(t+1) = Ax_i(t) + Bu_i(t), \tag{28}$$

where $A \in \mathbb{R}^{nxn}$ represents the autonomous dynamics, and $B \in \mathbb{R}^{nxm}$ represents the input dynamics. Specifically, we are concerned with double integrator dimension in two dimensions, thus $n = 4$ and $m = 2$.

The objective of the path planning is to minimize the control effort expended, which could also be considered fuel expenditure minimization. Control effort minimization could alternatively be replaced with a distance maximization or time minimization goal; however, it suffices to have these as constraints for the scope of this paper.

The vehicles must avoid a rectangular obstacle, defined by the intervals in each dimension $[X_{\min}X_{\max}], [Y_{\min}, Y_{\max}]$, maintaining a separation of d_{sep} in at least one dimension. As the obstacles later play a part in the determination of the channel capacity, they can be thought of as not only physical obstacles to avoid for collision avoidance purposes, but also as possible regions through which not to transmit, either to comply with emissions regulations or for secrecy reasons. In addition to avoiding obstacles, the vehicles must avoid colliding with each other, which is origin of the vehicle separation constraint.

The communication constraint for this path planning problem takes the form of an integral channel capacity requirement, a minimum on the sum of the channel capacity between over several timesteps. This constraint embodies the influence of the requirement for scheduling problem feasibility, in that it reflects the fact that any message passing must be completed in a limited amount of time, and duration of a message passing operation between vehicles is determined by the paths. Thus this constraint must be an integral constraint on the channel capacity between vehicles, as opposed to a pointwise constraint on the capacity, as seen in [5]. As the paths are allowed to "make up" the integral capacity, there are paths that are feasible with the integral constraint and thus the scheduling problem that would not be feasible with a pointwise constraint on the path planning. So, we introduce a constraint on on the path planning that is a function of the size of a message S, and the duration to send that message $d(S)$. For this, we need the concept of a conversion between message size and time. We start with

$$I(t_0, t_1) = \int_{t_0}^{t_1} C(\tau)d\tau, \tag{29}$$

which is the maxmimum information that be transmitted over a link during time interval $[t_0, t_1]$ with a given time-varying channel capacity $C(t)$. Thus, the earliest possible time t_c to complete the passing a message of size S, starting at time t_s can be found by solving

$$t_c = \min_{t \geq t_s} \quad t \tag{30}$$

$$s.t. \quad S \leq \int_{t_s}^{t} C(\tau)d\tau, \tag{31}$$

and we can introduce the constraint as

$$\int_{t}^{t+d(S)} C(\tau)d\tau \geq S \tag{32}$$

for all $t \in D$.

For computing channel capacity in this work, we use a simplified path loss model as found in Goldsmith [10], the time-dependent capacity of a channel is

$$C(t) = BW \log_2 \left(1 + \frac{P_t K}{N} \frac{1}{d^\gamma(t)}\right), \tag{33}$$

where $d(t)$ is the path length at time t, BW is the bandwidth of the channel, P_t is the transmit power, K is the free-space path loss w.r.t. one unit of d, and γ is the path-loss exponent. If the line-of-sight path is blocked by an obstacle, we treat the channel capacity as negligible at that point in time.

B. MIP formulation

We solve for the vehicle paths using a mixed-integer program, with a quadratic program for find a piecewise affine under-approximation of the channel capacity.

1) Decision variables: The decision variables for this program are the states x_i and inputs u_i for each vehicle $i \in V$, with an objective to minimize control effort with bounds on the instantaneous control, start positions, and end positions, so the program begins with:

$$\min \sum_{i}^{V} \sum_{t}^{T} ||u_i(t)||_2^2 \tag{34}$$

$$\text{s.t.} \quad u_{\min} \leq u_i(t) \leq u_{\max} \tag{35}$$

$$x_i(0) = x_{i,0} \tag{36}$$

$$x_i(T) = x_{i,f}, \tag{37}$$

where $x_{i,0}$ represents the starting position of vehicle i at time 0 and $x_{i,f}$ represents the final position of vehicle i at time t_f.

2) Dynamics: The double integrator dynamics are encoded by the constraints

$$x_i(t+1) = Ax_i(t) + Bu_i(t). \tag{38}$$

3) Obstacle avoidance: For encoding obstacle avoidance, we use the technique introduced by Schouwenaars with for a rectangular obstacle [6] . We introduce boolean variable, $b_{h,i}(t)$, are created to check each vehicle against the plane h of the obstacle, defined with constraints

$$X_i(t) - X_{max} \geq -Mb_{1,i}(t) \tag{39}$$

$$X_{min} - X_i(t) \geq -Mb_{2,i}(t) \tag{40}$$

$$Y_i(t) - Y_{max} \geq -Mb_{3,i}(t) \tag{41}$$

$$Y_{min} - Y_i(t) \geq -Mb_{4,i}(t) \tag{42}$$

where M is a large positive number, $X_i(t)$ is the position of vehicle i in the x, or first, dimension, and $Y_i(t)$ is the position of the vehicle in the y, or second, dimension. To enforce to avoidance of the obstacles, we add the constraint

$$\sum_{k=1}^{4} b_{k,i}(t) \leq 3, \tag{43}$$

that indicates the vehicle cannot be contained within more than three of the four planes of the rectangle.

4) Separation: In order to enforce the vehicle separation requirement, we replicate the obstacle avoidance, but with other vehicles. Accoringly, we introduce the binary decision variables $s_{i,j,h}(t)$, which are defined in the program using the constraints

$$X_i(t) - X_j(t) \geq d_{\text{sep}} - Ms_{i,j,1}(t) \quad \forall i \neq j \tag{44}$$

$$X_j(t) - X_i(t) \geq d_{\text{sep}} - Ms_{i,j,4}(t) \quad \forall i \neq j \tag{45}$$

$$Y_i(t) - Y_j(t) \geq d_{\text{sep}} - Ms_{i,j,2}(t) \quad \forall i \neq j \tag{46}$$

$$Y_j(t) - Y_i(t) \geq d_{\text{sep}} - Ms_{i,j,3}(t) \quad \forall i \neq j. \tag{47}$$

Using these binary decision variables, we enforce the separation with the constraint

$$\sum_{h=1}^{4} s_{i,j,h} \leq 3. \tag{48}$$

5) Piecewise affine channel capacity approximation: In order to constrain the channel capacity, we introduce a lower-bounding piecewise affine approximation on the channel capacity. Given an expected range of distances between vehicles, $[d_0, d_f]$, a number of piecewise segments to find W, a number of sample points D, and a set of indices for the samples associated with each segment S_w, we solve for affine coefficients to approximate each segment, $k_{1,i}, k_{2,i}$ for all $i \in W$, such that the approximation $\tilde{C}(d)$ is defined as:

$$\tilde{C}(d) = \begin{cases} k_{1,1}d + k_{2,1}, d \in S_1 \\ . \\ . \\ . \\ k_{1,W}d + k_{2,W}d, d \in S_W \end{cases} \tag{49}$$

We find these coefficients by solving a separate quadratic program for each segment $w \in W$

$$\min_{k_{1,w}, k_{2,w}} \quad \|\boldsymbol{e}\|_2^2 \tag{50}$$

$$\text{s.t.} \quad e_j = C(d_j) - k_{1,w}d_j + k_{2,w} \qquad \forall j \in S_w \tag{51}$$

$$e_j \geq 0 \qquad \forall j \in S_w, \tag{52}$$

where

$$C(d) = BW \log_2 \left(1 + \frac{P_t K}{N} \frac{1}{d^\gamma}\right). \tag{53}$$

6) Communication occlusion by obstacle: In order to account for an obstacle blocking the line-of-sight path of the communication channel, we introduce variables that signify if points sampled from along the path between the vehicles of interest are contained within an obstacle. For convenience, we use the term $s_{i,j,e,h}(t)$ to represent the position sample point e in spatial dimension h at time t of the link between vehicles i and j, which is defined as

$$s_{i,j,e,h}(t) = \frac{e}{E}(x_{i,h}(t) - x_{j,h}(t)) + x_{j,h}(t). \tag{54}$$

Given line search point e, obstacle o, face f of the obstacle, and time t, we construct the occlusion with the following for two dimensions: we first add the binary decision variables $c_{e,o,f}(t)$, defined using the constraint

$$s_{i,j,e,1}(t) - X_{max} \geq -Mc_{i,j,e,1}(t) \tag{55}$$

$$X_{min} - s_{i,j,e,1}(t) \geq -Mc_{i,j,e,2}(t) \tag{56}$$

$$s_{i,j,e,2}(t) - Y_{max} \geq -Mc_{i,j,e,3}(t) \tag{57}$$

$$Y_{min} - s_{i,j,e,2}(t) \geq -Mc_{i,j,e,4}(t) \tag{58}$$

We then introduce the binary decision variable, $l_{i,j,e}(t)$ that represents if sample point e on the link between vehicles i and j is within the obstacle at time t. In two spatial dimensions, this binary decision variable is constructed with two additional variables, $l_{i,j,e,1}$ and $l_{i,j,e,2}$, which represent the point being within the obstacle in the first dimension and

the second dimension respectively. The construction of $l_{i,j,e}(t)$ is enforced with the constraints

$$l_{i,j,e}(t) \geq l_{i,j,e,1} + l_{i,j,e,2} - 2 \quad (59)$$
$$l_{i,j,e}(t) \leq l_{i,j,e,1} \quad (60)$$
$$l_{i,j,e}(t) \leq l_{i,j,e,2}, \quad (61)$$

and the constructions of $l_{i,j,e,o,1}$ and $l_{i,j,e,o,2}$ are enforced with the constraints

$$l_{i,j,e,1}(t) \geq c_{i,j,e,1}(t) + c_{i,j,e,2}(t) - 1 \quad (62)$$
$$l_{i,j,e,1}(t) \leq c_{i,j,e,1}(t) \quad (63)$$
$$l_{i,j,e,1}(t) \leq c_{i,j,e,2}(t) \quad (64)$$

and

$$l_{i,j,e,2}(t) \geq c_{i,j,e,3}(t) + c_{i,j,e,i,4}(t) - 1 \quad (65)$$
$$l_{i,j,e,2}(t) \leq c_{i,j,e,3}(t) \quad (66)$$
$$l_{i,j,e,2}(t) \leq c_{i,j,e,4}(t) \quad (67)$$

respectively. We then add a binary variable, $o_{i,j}(t)$, to identify if the the line-of-sight path between vehicles i and j is obstructed at time t, that is constructed as a logical and of all of the $l_e(t)$ at a give time t across all points and obstacles.

7) Pointwise channel capacity definition: With the occlusion variable $o_{i,j}(t)$ and the piecewise affine lower-bounding approximation coefficients, we construct a pointwise channel capacity expression in the mixed-integer program using two additional decision variables. The first is a binary decision variable that represents is the current distance between vehicle i and vehicle j is within piecewise segment w at time time t, $q_{i,j,w}(t)$, and the other scalar to hold the value of the distance if it is in segment w, $r_{i,j,w}(t)$. These decision variables are defined in the program by the constraints

$$b_{i,j}(t) + \sum_{w=1}^{W} q_{i,j,w}(t) = 1 \quad (68)$$
$$\sum_{w=1}^{W} r_{i,j,w}(t) \leq d_{i,j}(t) \quad (69)$$
$$r_{i,j,w}(t) \leq H(w) q_{i,j,w} \quad (70)$$
$$r_{i,j,w}(t) \geq H(w-1) q_{i,j,w}. \quad (71)$$

We then define an expression $a_{i,j}(t)$ for the value of the piecewise approximation of the capacity between vehicles i and j at time t taking into account obstacles as follows:

$$a_{i,j}(t) = \sum_{w=1}^{W} k_{1,w} r_{i,j,w}(t) + k_{2,w} q_{i,j,w}(t). \quad (72)$$

8) Integral channel capacity constraint: With the pointwise approximation expression defined, we can define the summation channel capacity contraint with the program constraint

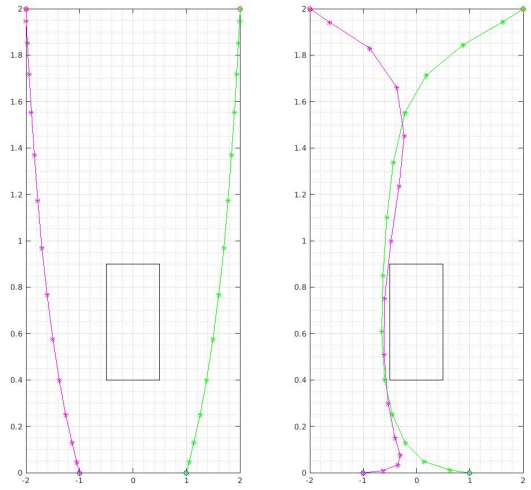
$$\sum_{t}^{t+d} a_{i,j}(t) \geq P \quad \forall t \in T - d, \quad (73)$$

T(sec)	dt(dec)	$\|u\|$	BW(kHz)	PK/N	γ	d_{sep}(m)
12	0.75	0.5	0.01	1	2	0.1

TABLE I: Parameters of example seen in Figure 3 of MIP for path planning.

where P is the size of payload and d is the duration of the constraint in terms of the number of time-steps to provide the This concludes the program.

For an example of the impact of this constraint, two paths, one with a negligible and one with a non-neglible communication load constraint, can be seen in Figure 3, and the values to produce this can be seen in Table I.



(a) Example of insignificant message size constraint, payload size requirement of $P = 0.01$ kb.

(b) Example of significant message size constraint, a payload size requirement of $P = 0.1$ kb.

Fig. 3: Example of MIP results for multi-vehicle path planning with integral communication capacity constraints.

C. Combining scheduling and path planning

The path planning method is communication-aware, but the correct values for the communication constraints are necessary to ensure a path is generated that yields a feasible schedule. As the schedule improves with lower communication times for higher payloads, and the path is further constrained, the performance of the two methods is complementary.

In combining the scheduling and path planning, we aim to return the optimal path, in the sense that it has the lowest path objective value constrained by the scheduling problem being feasible. For this purpose, we do a search on the the payload size parameter of the constraint, P. In this search, if both methods are feasible, the constraint is loosened until the scheduling problem is no longer feasible, and the last known payload size for which both methods are feasible. If the constraint is too strong to yield a feasible path from the

path planning MIP, and too weak to enable a feasible schedule, then the overall problem is infeasible.

In order to initially compute the communication constraints, we subtract the sum of the durations of the processing tasks first. From this value we divide by the number of communication tasks, and then set this to be the constraint for capacity with the largest of the communication tasks. This approach is conservative, but yields a close initial solution, which decreases the planning time.

IV. Case study

A. Case study scenario

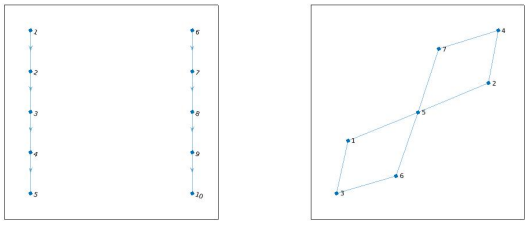
Let us examine the use of this the scheduling and path planning methods presented for application to groups of vehicles that utilize offloading to enable tolerance of onboard computing hardware failure. Consider the case of two identical synchronized vehicles, each with a safety-critical application specific to that vehicle that share a common application cycle and path horizon. In this case, we consider fault-tolerance to be the ability of safety-critical applications to be scheduled and processed over the next planning and execution cycle. To demonstrate fault-tolerance, we will demonstrate that the combining the scheduling and path planning produces a solution when both processors are available, as well as when only one is available.

In the event of a fault, we assume the failing vehicle enters a detectable fail-safe mode until the next planning and scheduling period. Also, the failure of the computing hardware must not lead to a complete failure of the vehicle implicitly, e.g. a high-power processor that does not handle low-level control, communication, or vehicle management and is instead used for intensive applications such as perception and optimization. As all vehicles have access to the schedules, if the vehicle that failed was responsible for computing the next centralized planning phase, then another vehicle could restart the cycle as the initial scheduler. In the case we examine, the process of computing the combined schedule and path is not itself scheduled, but if it were to, it would take the form of the application precedence graph as seen if Figure 2b.

B. Case study simulation and results

For simulating the specific case, we used two applications composed of five tasks each, with precedence relations as seen in Figure 4a. These were to be scheduled on two vehicles, each having a nominally operational processor with a loopback and access to the shared communication channel. From this, we modelled the overall processor network as seven processors, with two ownership machines, three communication machines, and two processing machines; this network can be seen in Figure 4b.

For theses parameters, we generated two sets of paths and schedules. The first case, whose path can be seen in 5a and schedule in Figure 6a, is the nominal operational case. The second case, whose path can be seen in 5b and schedule in Figure 6b, is the failure case, in which one of the processing machines is removed from the scheduling problem. The only parameters modified were the removal of this machine; all other changes are a result of the effects of this removal enforced by the resulting method from combining the scheduling and path planning.



(a) Task precedence graph for case study.

(b) Machine precedence graph for case study.

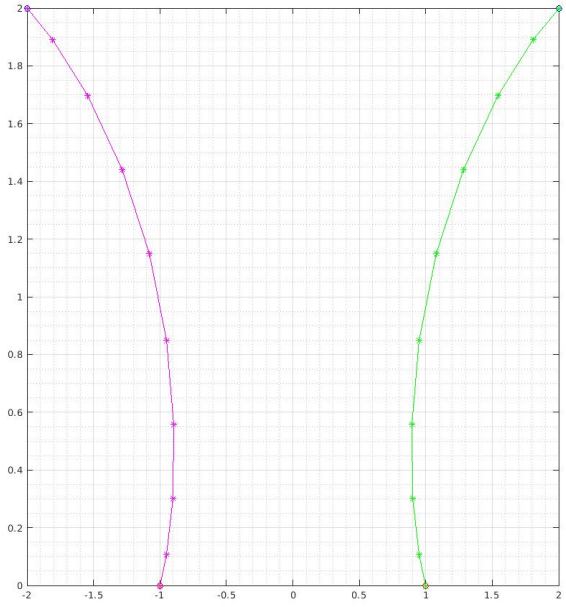
Fig. 4: Precedence graphs for case study applications.

V. Conclusion

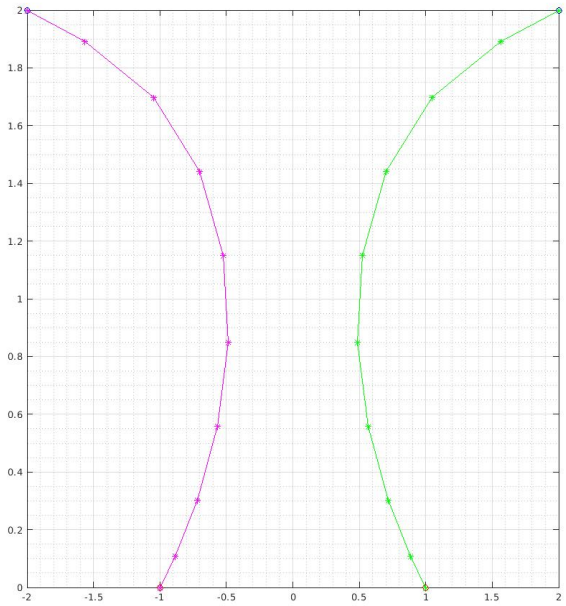
We present a scheduling method, a path-planning method, and a method that combines the two for use with offloading applications within groups of networked vehicles. For the scheduling method and path-planning, we describe the problem and provided details on the mixed-integer program we use to solve them. Finally, we demonstrate these schemes with a case study of two vehicles demonstrating how it may be used to enable fault-tolerance with respect to the onboard high-powered processor. In the case study, the resulting paths and schedules demonstrate the methods, when combined, could account for the changes in the problem in the case the removal of one of the processors, i.e. a fault.

In conducting this work and simulation for the case study, the overall architecture is implemented in MATLAB. The path planning MIP is implemented using CVX with MOSEK as the solver [11][12]. The scheduling MIP is implemented using Python-MIP with COIN-OR Cbc as the backend solver [13][14].

There are several possible improvements to this architecture. The scheduling and path planning programs both involve large numbers of integer variables, and the programs do not scale well, and a more efficient formulation or approximation algorithm would be required for practical use. Additionally, the efficiency of the method for combining scheduling and path planning could be improved, or made constant time, by further analysis of the feasibility of scheduling for offloading in order to find better initial constraints, or solve and search better for constraints. Finally, it might be possible to combine the two methods into one MIP, eliminating the need for a method to combine the scheduling and path planning in an iterative manner, and addressing the efficiency deficiencies of the first method. Nonetheless, the methods demonstrate in this paper are sufficient for finding feasible paths and schedules in the the context of offloading for fault-tolerance.

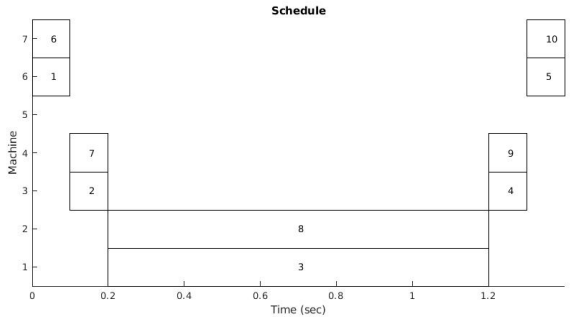


(a) Vehicle paths for the nominal case.

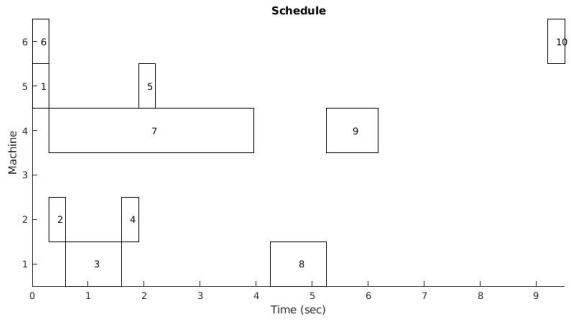


(b) Vehicle paths for the failure case.

Fig. 5: Vehicle paths for the nominal and failure cases. Note how the path for the failure case is narrower that for the nominal case in order to provide the necessary channel capacity to support the offloading of the safety-critical application of the vehicle with a failure.



(a) Schedule for the nominal case.



(b) Schedule for the failure case.

Fig. 6: Schedule of application related tasks for the nominal and failure cases. Note how for the schedule of the failure case the inter-vehicle communication channel machine is now utilized in order to communicate the input and output of the safety-critical application if the vehicle with a failure.

REFERENCES

[1] D. Wang, Z. Liu, X. Wang, and Y. Lan, "Mobility-aware task offloading and migration schemes in fog computing networks," *IEEE Access*, vol. 7, pp. 43 356–43 368, 2019. DOI: 10.1109/access.2019.2908263. [Online]. Available: https://doi.org/10.1109%2Faccess.2019.2908263.

[2] S. A. Zanlongo, A. C. Wilson, L. Bobadilla, and T. Sookoor, "Scheduling and path planning for computational ferrying," in *MILCOM 2016-2016 IEEE Military Communications Conference*, IEEE, 2016, pp. 636–641.

[3] S. Gertphol and V. Prasanna, "MIP formulation for robust resource allocation in dynamic real-time systems," in *Proceedings International Parallel and Distributed Processing Symposium*, IEEE Comput. Soc. DOI: 10.1109/ipdps.2003.1213231. [Online]. Available: https://doi.org/10.1109%2Fipdps.2003.1213231.

[4] I. Hamaz, L. Houssin, and S. Cafieri, "The Cyclic Job Shop Problem with uncertain processing times," in *16th International Conference on Project Management and Scheduling (PMS 2018)*, Rome, Italy, Apr. 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01876486.

[5] E. I. Grøtli and T. A. Johansen, "Path planning for uavs under communication constraints using splat! and milp," *Journal of Intelligent & Robotic Systems*, vol. 65, no. 1-4, pp. 265–282, 2012.

[6] T. Schouwenaars, A. Stubbs, J. Paduano, and E. Feron, "Multivehicle path planning for nonline-of-sight communication," *Journal of Field Robotics*, vol. 23, no. 3-4, pp. 269–290, 2006.

[7] A. Ghaffarkhah and Y. Mostofi, "Communication-aware motion planning in mobile networks," *IEEE Transactions on Automatic Control*, vol. 56, no. 10, pp. 2478–2485, 2011. DOI: 10.1109/tac.2011.2164033. [Online]. Available: https://doi.org/10.1109%2Ftac.2011.2164033.

[8] E. Balas, "Disjunctive programming and a hierarchy of relaxations for discrete optimization problems," *SIAM Journal on Algebraic Discrete Methods*, vol. 6, no. 3, pp. 466–486, 1985.

[9] T. Davidovic, L. Liberti, N. Maculan, and N. Mladenovic, "Towards the optimal solution of the multiprocessor scheduling problem with communication delays," 2007.

[10] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[11] M. Grant and S. Boyd, *CVX: Matlab software for disciplined convex programming, version 2.1*, http://cvxr.com/cvx, Mar. 2014.

[12] Mosek, "The mosek optimization software (2020)," *URL http://www.mosek.com*, 2020.

[13] H. Santos and T. Toffolo, *Python mip: Version 1.8.1*, Apr. 2020.

[14] J. Forrest, T. Ralphs, S. Vigerske, LouHafer, B. Kristjansson, jpfasano, EdwinStraver, M. Lubin, H. G. Santos, rlougee, and M. Saltzman, *Coin-or/cbc: Version 2.9.9*, version releases/2.9.9, Jul. 2018. DOI: 10.5281/zenodo.1317566. [Online]. Available: https://doi.org/10.5281/zenodo.1317566.