IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems, 29 Oct./2 Nov., San Diego, CA, 2007

A Decentralized Planner that Guarantees the Safety of Communicating Vehicles with Complex Dynamics that Replan Online

Kostas E. Bekris Konstantinos I. Tsianos Lydia E. Kavraki

***Abstract*— This paper considers the problem of coordinating multiple vehicles with kinodynamic constraints that operate in the same partially-known environment. The vehicles are able to communicate within limited range. Their objective is to avoid collisions between them and with the obstacles, while the vehicles move towards their goals. An important issue of real-time planning for systems with bounded acceleration is that inevitable collision states must also be avoided. The focus of this paper is to guarantee safety despite the dynamic constraints with a decentralized motion planning technique that employs only local information. We propose a coordination framework that allows vehicles to generate and select compatible sets of valid trajectories and prove that this scheme guarantees collision-avoidance in the specified setup. The theoretical results have been also experimentally confirmed with a distributed simulator where each vehicle replans online with a sampling-based, kinodynamic motion planner and uses message-passing to communicate with neighboring agents.**

I. Introduction

Generating safe and effective motions for realistic automobiles and autonomous mobile platforms is an important application area of motion planning. Using multiple, coordinating vehicles can offer redundancy and robustness in the execution of many tasks (e.g. space exploration, autonomous demining). Often, both the motivation and the primary concern for deploying such systems in everyday tasks is the issue of safety. One way to achieve safe coordination is to employ inter-vehicle communication so as to avoid collisions and improve the efficiency of the planned motions. This realization has led recently to the active development of vehicle-to-vehicle communication technology [18].

This work studies the problem of generating collision-free motions for multiple vehicles operating in the same, partially-known space (see Fig. 1). The vehicles obey kinodynamic constraints (e.g. bounded acceleration) and they communicate using wireless networking if they are within range. The objective is to avoid collisions between vehicles and with workspace obstacles, despite the dynamics. The focus is on decentralized, online solutions, where each vehicle plans only in its own state-space and recomputes trajectories online. This paper extends replanning algorithms for a vehicle with complex dynamics [15], [1] to a decentralized framework for multiple communicating vehicles. We provide a theoretical study on the sufficient requirements that decentralized methods must satisfy to guarantee safety.

Work on this paper by K. Bekris, K. Tsianos and L. Kavraki has been supported in part by NSF 0308237, 0615328 and 0713623. The computational experiments were run on equipment obtained by CNS 0454333, and CNS 0421109 in partnership with Rice University, AMD and Cray. The authors are with the Computer Science Department, Rice University, Houston, TX, 77005, USA {bekris,konstantinos,kavraki}@rice.edu

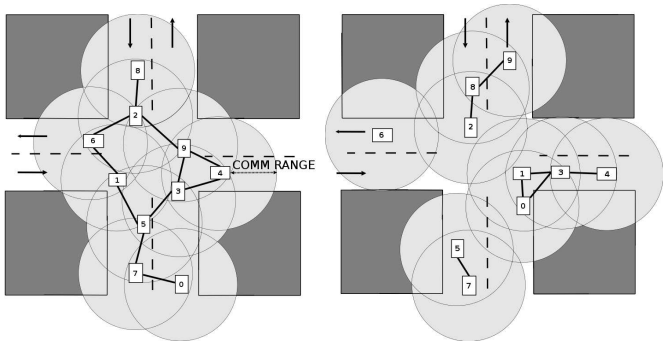

Fig. 1. Vehicles form a communication network while they move. On the left, there is one connected component while on the right vehicles have moved and multiple components have been created. Planning for such dynamic networks with centralized approaches has been studied for first-order systems [6], [7]. This paper extends these ideas by considering second order dynamics (we guarantee avoidance of ICS) and describing a decentralized solution using only local information.

The proposed approach enforces the invariant that every vehicle has at least one collision-free plan available at each replanning cycle. This can be achieved using only local communication with neighboring vehicles and without conservative, worst-case assumptions about the motion of other vehicles [4]. Given the exchanged information, each vehicle invokes a sampling-based, kinodynamic planner. Thus, vehicles avoid planning in the product space as in centralized methods [17], [6], and select the best trajectory given their individual goal. Although planning is decentralized in our method, we provide a proof that the proposed algorithm guarantees collision-avoidance between vehicles and with static obstacles and also prevents vehicles from ending up in inevitable collision states (ICS) [9], [15]. This result is also extended to teams of vehicles that have to retain a communication network.

The planner has been implemented on a multi-processor, message-passing simulator that models multiple vehicles. Each processor models a vehicle, communicates asynchronously with other processors and replans online. Our experiments confirm the theoretical result that the proposed scheme guarantees safety; something that simplistic prioritized coordination schemes cannot.

A. Background

Kinodynamic Planning: Planning for one vehicle is already challenging and complicated by the presence of kinodynamic constraints, such as drift. Sampling-based approaches, like tree-based planners [14], [11], [13], have been successful in dealing with dynamic constraints.

Replanning: Planning with partial-observability requires interleaving sensing, planning and execution, where a planner is called frequently and has finite time to replan a trajectory. Replanning from scratch is possible [11] but recent methods use information from previous planning cycles to speed up the performance of replanning [5], [8], [19], [10]. There are also methods that use a previously constructed roadmap to replan online [4], [12].

Safety: Replanning with bounded acceleration raises safety concerns since a robot can reach an `ICS` [9], [15]. An extension of basic sampling-based planners avoids `ICS` at a low computational cost [1].

Coordination: Centralized planning approaches have been proposed for coordinating dynamic networks which form when vehicles operate in the same area [6], [7]. *Centralized* methods are reliable [17] but also computationally expensive due to the exponential dependency of motion planning on problem dimensionality. *Decentralized* methods, such as prioritized schemes [3], plan separately for each robot and then coordinate the robots' interactions. Planning can be orders of magnitudes faster but can also lead to collisions [17], [6]. An important research direction is how to make decentralized approaches more reliable [16].

B. Contribution

In relation to the existing literature, this paper:

- Extends work on `ICS` [9], [1] to the case of multiple, coordinating vehicles.
- Extends planning methods for dynamic networks [6] to problems with more complicated dynamics and proposes a fully decentralized framework.
- Provides a theoretical analysis that decentralized replanning can be not only fast but also safe [17], [16]. Results from a distributed simulation reaffirm the analysis.

II. Problem Setup

Consider vehicles $V_1, \ldots, V_v$ deployed in the same partially-known environment, trying to move towards their individual goals. A vehicle's motion is governed by the differential equations: $\dot{x}(t) = f(x(t), \alpha)$ and $g(x(t), \dot{x}(t)) \leq 0$, where $x(t) \in \mathcal{X}$ represents a state, $\alpha \in \mathcal{A}$ is a control, f, g are smooth and t is time. This paper focuses on systems with bounds in velocity and acceleration. Each vehicle is equipped with wireless communication capabilities. When two robots are within range they establish a communication link.

How can the vehicles communicate so as to guarantee collision avoidance despite kinodynamic constraints? Is it possible to obtain a decentralized solution where each vehicle needs only local information for planning?

The following assumptions are being made in this work:

- Communication is reliable, offers sufficient bandwidth and is not affected by line of sight constraints. The vehicles synchronize their operation.
- We do not deal with issues related to uncertainty. We assume that motion commands selected and communicated by a vehicle are executed fairly accurately and there are no sensing errors.

Notation:

- A state $x_i(t)$ is **collision-free** if V_i does not collide with obstacles. If vehicles V_i, V_j are not in collision at t, their states $x_i(t), x_j(t)$ are **compatible states**: $x_i(t) \asymp x_j(t)$.
- A **plan** $p(dt) = \{(\alpha_1, dt_1)), \ldots, (\alpha_n, dt_n)\}$ is a time sequence of controls with duration $dt = \sum_i dt_i$.
- A **trajectory** $\pi(x(t), p(dt))$ is the resulting sequence of states when a plan $p(dt)$ is executed at state $x(t)$. A trajectory is **feasible** if it respects f and g. A plan $p(dt)$ is **valid** at state $x(t)$ if it produces a feasible trajectory $\pi(x(t), p(dt))$.
- State $x^{\pi}(t')$ occurs along trajectory π at time t'. A **collision-free trajectory** must be feasible and:

$$\forall\ t' \in [t : t + dt] :\ x^{\pi}(t') \text{ is collision-free.}$$

- Trajectories $\pi_i(x_i(t_i), p_i(dt_i))$ and $\pi_j(x_j(t_j), p_j(dt_j))$ are **compatible trajectories** ($\pi_i \asymp \pi_j$) iff:

$$\forall\ t' \in [\max\{t_i, t_j\} : \min\{t_i + dt_i, t_j + dt_j\}] : \\ x^{\pi_i}(t') \asymp x^{\pi_j}(t').$$

- **Trajectory concatenation** $\pi'(\ \pi(x(t), p(dt)),\ p'(dt')\ )$ is the sequence of states a vehicle follows when it first executes the plan $p(dt)$ at state $x(t)$ and after the completion of $p(dt)$ the vehicle executes plan $p'(dt')$.

III. Decentralized Coordinated Planning

We first describe a planner that a single vehicle can use to replan and avoid `ICS`. We then extend this algorithm to the case of multiple communicating vehicles, initially assuming that all vehicles can communicate. This assumption is later waived and the algorithm is also extended to the case of a vehicular network.

A. Safe Motion Planning for a Single Vehicle

The algorithm incrementally expands a tree data structure in the vehicle's state-time space and returns safe paths given a partially-known workspace and kinodynamic constraints [1]. There are two elements of the approach relevant to the multi-robot problem: (i) The planner's operation is broken into consecutive replanning cycles. (ii) Within a cycle, the planner avoids not only collisions but also avoids `ICS`.

During cycle $(t_{n-1} : t_n)$, the planner uses an updated model of the world up to time t_{n-1} and an estimate of the state $x(t_n)$ at the beginning of the next planning cycle $(t_n : t_{n+1})$. Given a goal, the planner computes a new plan before t_n that will be executed during the next cycle: (t_n, t_{n+1}), as in Fig. 2 (left). This is achieved by expanding a tree data structure $(Tree)$ in the vehicle's state-time space using a sampling-based approach [1], [14], [11], [13]. From the expanded tree, a valid plan $p(t_n : t_{n+1})$ that results in the trajectory $\pi(x(t_n), p(t_n : t_{n+1}))$ must be selected.

It is not sufficient for $\pi(x(t_n), p(t_n : t_{n+1}))$ to be just collision-free, since it may lead to an `ICS` [9], as Fig. 2 (right) demonstrates. It is computationally intractable, however, to check if a state is truly `ICS` or not: all possible plans out of that state have to be examined to determine if there is an escape plan. It is sufficient, however, to take a

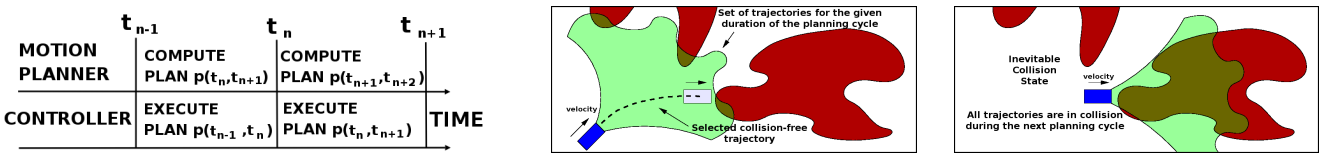

Fig. 2. (left) The robot's synchronization scheme. (center and right) A valid plan may still lead to an ICS during the next planning cycle.

conservative approach: if the vehicle can avoid collisions by executing a pre-specified "contingency" plan $\gamma(\cdot)$ out of a state x, then x is safe. In other words, state x is **safe** iff:

$$\exists\ \gamma(\infty) \text{ s.t. } \pi(x, \gamma(\infty)) \text{ is collision free.} \quad (1)$$

In our simulation, the contingency plan we use for car-like vehicles is a breaking maneuver that brings the car to a complete stop as fast as possible. The duration of a contingency plan depends on the vehicle's velocity at state x and its acceleration bound. Since the planner is required to return a plan only for the period $(t_n : t_{n+1})$, only the states along the tree that occur at time t_{n+1} have to be checked whether they are safe or not. The planner implements the following invariant for all plans $p(t_n : t_{n+1})$ along the tree:

$$\pi(\ \pi(x(t_n), p(t_n : t_{n+1})),\ \gamma(\infty)\ ) \text{ is collision-free.}$$

This means that for all plans $p(t_n : t_{n+1})$ there is a concatenation with contingency plans that leads to collision-free trajectories. With this method, a vehicle can operate in a partially-known workspace with static obstacles and can avoid collisions at all times [1].

B. Safe Coordinated Planning: Unlimited Communication

We now move on to the case of multiple vehicles in the environment executing the same replanning loop. This section deals with vehicles that have unlimited communication range. Section III-C waives this assumption.

We will first describe a simple extension of the single-vehicle algorithm to a coordinated approach. As Fig. 1 shows, communication links between vehicles define a graph, where the vehicles are nodes and two vehicles share an edge if the two vehicles can exchange messages. In the case of unlimited communication range this graph is complete. Suppose every vehicle has a unique global priority. We define the set N^h to represent the neighbors of vehicle V on the communication graph with higher priorities than V and the set N^l to be the set with lower priorities. Then the simple prioritized scheme executed on each vehicle V during a single planning cycle $(t_n : t_{n+1})$ has the following step:

1) Compute a set of candidate plans $\mathcal{P}$ of duration $(t_n : t_{n+1})$ with the single-vehicle algorithm.
2) Receive the selected plans $\mathcal{P}^h$ from neighbors in N^h.
3) Select plan $p(t_n : t_{n+1}) \in \mathcal{P}$ that does not collide with plans in $\mathcal{P}^h$ and best serves the goal of vehicle V.
4) Transmit the plan p to all neighbors N^l.

The simple extension, however, fails to produce safe trajectories for multiple reasons:

- If a cycle is completed before all higher priority plans are received, no plan p can be safely selected.
- Even if $p \in \mathcal{P}$ and $\mathcal{P}^h$ are available on time, it may be that no plan p is collision-free with all plans in $\mathcal{P}^h$ due to the decentralized nature of the approach.
- Suppose p is collision-free with set $\mathcal{P}^h$. It may still lead to ICS given $\mathcal{P}^h$ due to the dynamics.

The definition of a safe state from Eq. 1 is inadequate in the multi-vehicle case, where the safety of a vehicle's state depends on the states and the choices of the other vehicles. We extend the definition of safety as follows:

Safe State - Multi-vehicle case: Consider vehicles $V_1, \ldots, V_v$ that have states $x_1(t), \ldots, x_v(t)$ and all vehicles $V_j, j \neq i$ execute plans $p_j(dt)$. Then state $x_i(t)$ is safe iff $\exists\ \gamma_i(\infty)$ so that:

$$\pi_i(x_i(t), \gamma_i(\infty)) \text{ is collision free} \quad \wedge \quad \forall\ j \neq i: \\ \pi_i(x_i(t), \gamma_i(\infty)) \asymp \pi'_j(\ \pi_j(x_j(t), p_j(dt)),\ \gamma_j(\infty)\ ) \quad (2)$$

Note that the trajectory $\pi_i(x_i(t), \gamma_i(\infty))$ must be compatible with the concatenation of other vehicles' plans and contingencies. Given this new definition of a safe state, we set an objective for the coordination algorithm we described earlier. It must satisfy the following.

Invariant: For each replanning cycle $(t_n : t_{n+1})$ every vehicle V_i selects a plan $p_i(t_n : t_{n+1})$ which when executed at state $x_i(t_n)$:

a. The resulting trajectory $\pi_i(x_i(t_n), p_i(t_n : t_{n+1}))$ is collision-free.
b. During the current cycle $(t_n : t_{n+1})$, it is compatible with all other vehicles, $\forall j \neq i$:

$$\pi_i(x_i(t_n), p_i(t_n : t_{n+1})) \asymp \pi_j(x_j(t_n), p_j(t_n : t_{n+1})).$$

c. It leads to state $x^{\pi_i}(t_{n+1})$ that is safe according to Eq. 2 for every choice of plans $p_j(t_{n+1} : t_{n+2})$ that the other vehicles may make during the next planning cycle.

If the Invariant holds then the algorithm will produce safe trajectories. Points a. and b. imply that there is no collision during the current cycle $(t_n : t_{n+1})$, either with static geometry or between vehicles. Point c. implies that all vehicles at the next cycle $(t_{n+1} : t_{n+2})$ have contingency plans which can be followed regardless of the other vehicles' choices. Consequently, the prioritized algorithm in the beginning of this section can be altered so that step 3 is:

3) Select plan $p(t_n : t_{n+1}) \in \mathcal{P}$ that satisfies the Invariant given the set P^h. If no such plan exists or time is running out, execute contingency $\gamma(t_n : t_{n+1})$, which is precomputed from the previous planning cycle and collision-free due to the Invariant.

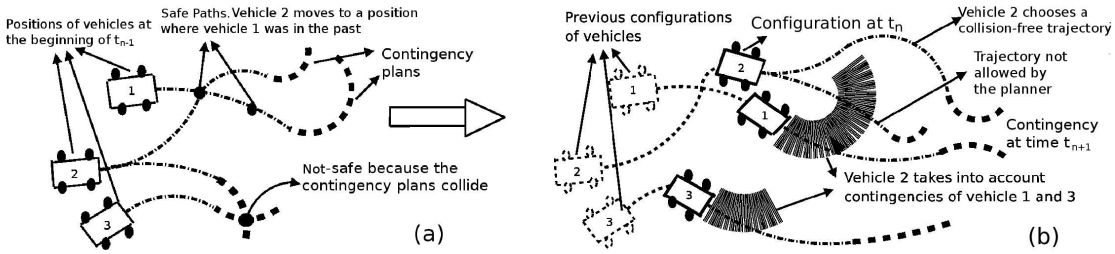

Fig. 3. (a) The lower plan for V_2 is not safe since the contingency attached to it collides with the contingency extending from the plan of V_3. The top plan of V_2 is safe. (b) The planner of V_2 will not produce the lower trajectory because it collides with the current contingency of V_1. The top plan is again safe.

Consequently, now we need to answer the question of: *how to produce and select plans $p(t_n : t_{n+1})$ that satisfy the Invariant*. We propose that any selected plan at step 3 of the algorithm must satisfy:

Requirement 1: As in the single-vehicle case, the concatenation of plan $p_i(dt)$ with a contingency plan $\gamma_i(\infty)$ must be collision-free:

$$\pi'(\ \pi(x(t_n), p(t_n : t_{n+1})),\ \gamma(\infty)\ ) \text{ is collision-free.} \quad (3)$$

Requirement 2: The concatenation of plan $p_i(dt)$ with a contingency plan $\gamma_i(\infty)$ must be compatible with the contingency plans $\gamma_j(\infty)$ of other vehicles:

$$\forall j \neq i: \quad \pi_i'(\ \pi_i(x_i(t_n), p_i(dt)), \gamma_i(\infty)) \succ\ \pi_j(x_j(t_n), \gamma_j(\infty)) \quad (4)$$

Requirement 3: The concatenation of plan $p_i(dt)$ with a contingency plan $\gamma_i(\infty)$ must be compatible with the concatenations of plans $p_j(dt)$ of other vehicles with their contingency plans $\gamma_j(\infty)$:

$$\forall j \neq i: \quad \pi_i'(\ \pi_i(x_i(t_n), p_i(dt)), \gamma_i(\infty)) \succ \pi_j'(\ \pi_j(x_j(t_n), p_j(dt)), \gamma_j(\infty)) \quad (5)$$

Theorem: Assume the Invariant is satisfied during planning cycle $(t_{n-1} : t_n)$ for all vehicles. Then if each vehicle V_i selects a plan $p_i(t_n : t_{n+1})$ that satisfies Eq. 3, 4 and 5 or selects an available contingency plan, then the Invariant will also hold during the next planning cycle $(t_n : t_{n+1})$.

Proof: We will have to show that the three points of the Invariant are satisfied during the next planning cycle $(t_n : t_{n+1})$. There are two cases. Either the algorithm manages to produce and select a plan $p_i(t_n : t_{n+1})$ that satisfies Eq. 3, 4 and 5 or selects a contingency plan. We will treat these two cases separately:

1) Assume such plan $p_i(t_n : t_{n+1})$ has been found. Because the plan satisfies Eq. 3 and Eq. 5, points a. and b. of the Invariant are satisfied, respectively. Point c. is more complicated. The state $x(t_{n+1})$ that the vehicle will reach after executing $p_i(t_n : t_{n+1})$ must have the property that it is safe according to Eq. 2. The application of the contingency plan γ_i at state $x(t_{n+1})$ will result in a collision-free path according to Eq. 3, so one of the two specifications of Eq. 2 is satisfied. State $x(t_{n+1})$ has to be safe, however, for any choice of plans $p_j(t_{n+1} : t_{n+2})$ that the other vehicles will make during the next planning cycle. There are again two possible cases for the nature of plans another vehicle V_j can follow during cycle $(t_{n+1} : t_{n+2})$:

a. Assume vehicle V_j computes a plan $p_j(t_{n+1} : t_{n+2})$ that satisfies the requirements. Then due to Eq. 4, this plan is compatible with the contingency of V_i during that cycle:
$$\pi_i(x^{\pi_i}(t_{n+1}), \gamma_i(\infty)) \succ \pi_j'(\ \pi_j(x^{\pi_j}(t_{n+1}), p_j(t_{n+1} : t_{n+2})),\ \gamma_j(\infty)\ )$$

b. Assume vehicle V_j resorts to a contingency during cycle $(t_{n+1} : t_{n+2})$. Due to Eq. 5, however, the contingency of V_j is by construction compatible with the contingency of V_i:
$$\pi_i(x^{\pi_i}(t_{n+1}), \gamma_i(\infty)) \succ \pi_j(x^{\pi_j}(t_{n+1}), \gamma_j(\infty))$$

In any case, Eq. 2 is satisfied for state $x^{\pi_i}(t_{n+1})$, which means that the third point of the Invariant is also satisfied for the next planning cycle.

2) Assume that vehicle V_i has to resort to a contingency. The inductive hypothesis is that the Invariant holds during the current cycle, so the state $x(t_n)$ is safe according to Eq. 2 for every choice of plans of other vehicles. From Eq. 2 the points a. and b. of the Invariant trivially hold for the trajectory that follows the contingency plan. In order to show that the state $x(t_{n+1})$ reached after the application of the contingency plan $\gamma_i(t_n : t_{n+1})$ is safe according to Eq. 2 we can follow exactly the same reasoning as above. From Eq. 3 the trajectory $\pi_i(x(t_{n+1}), \gamma_i(\infty))$ will be collision-free and will also be compatible given any choice the other vehicles will make due to Eq. 4 and 5. □

Algorithm

We describe here how Algorithm 1 satisfies the requirements with a priority scheme. Fig. 3 provides an illustration of the algorithms operation. For the second requirement, each vehicle V_i must be aware of the contingencies of other vehicles V_j at state $x(t_n)$ during planning cycle $(t_{n-1} : t_n)$. These contingencies have been computed by each V_j during the previous step and can be communicated at the beginning of each cycle. After exchanging contingencies, the sampling-based, kinodynamic planner is invoked. It generates a tree of feasible trajectories in the state-space that are collision-free and avoids `ICS` with obstacles in the beginning of the consecutive cycle (Eq. 3). The planner considers also in

Algorithm 1 COORDINATED ICS AVOIDANCE for V_i

Identify set of neighbors $N = N^h \cup N^l$

`(Exchange contingencies)`
for all $j \in N$ **do**
  Send contingency $\gamma_i(\infty)$ to V_j
  Receive contingency $\gamma_j(\infty)$ from V_j

`(Planning: satisfies requirements 1,2)`
$HN \leftarrow N^h$ (high priority neighbor set)
Select $PlanningBudget$ according to priority
$Tree \leftarrow$ Retain valid subset of $Tree$ from previous cycle
while $(time < PlanningBudget) \wedge HN \neq \emptyset$ **do**
{
  `(Sampling-Based Kinodynamic Planning)`
  Select an existing trajectory sample s from $Tree$
  Select plan $p(dt)$ and state $x(t)$ on s
  Propagate trajectory $\pi(x(t), p(dt)$

  `(Req. 1: Avoid ICS with obstacles)`
  if ($\pi(x(t), p(dt))$ is not **collision-free**) **then**
    Reject π
  else
    if $(t < t_{n+1}) \wedge (t + dt > t_{n+1})$ **then**
      `(path intersects next cycle` t_{n+1}`)`
      if ($\pi(\pi(x(t), p(dt), \gamma(\infty))$ not collision-free) **then**
        `(Leads to ICS with obstacles)`
        Reject π

  `(Req. 2: Compatibility with` $\gamma_j(\infty)$`)`
  for all $j \in N$ and while π is not rejected **do**
    if ($\pi(x(t), p(dt)) \not\succ \pi_j(x_j(t_n), \gamma_j(\infty))$) **then**
      `(Does not respect Eq. 4)`
      Reject π

  `(Receive high priority plans)`
  if (message arrived from $j \in HN$) **then**
    Receive selected plan $p_j^*(t_n : t_{n+1})$
    Receive contingency $\gamma_j^*(\infty)$ at $x_j(t_{n+1})$
    Remove j from HN
}

`(Path Selection: satisfies req. 3)`
$p_i^* \leftarrow \gamma_i(\infty)$ (safe from previous round)
$P' \leftarrow$ Extract all plans $p_i'(t_n : t_{n+1})$ from $Tree$
for all $p_i' \in P'$ and while $(time < PlanningCycle)$ **do**
  for all $j \in N^h$ **do**
    if (Eq. 5 does not hold for $p_i', p_j^*(t_n : t_{n+1}), \gamma_j^*(\infty)$) **then**
      Reject p_i'
  if p_i' is not rejected and p_i' better than p_i^* **then**
    $p_i^* \leftarrow p_i'(t_n : t_{n+1})$

`(Transmit selected plan)`
for all $j \in N^l$ **do**
  Send selected plan p_i^* to V_j
  Send contingency $\gamma_i^*(\infty)$ at $x_i(t_{n+1})$ to V_j

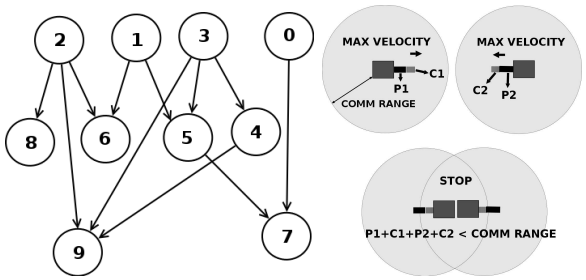

Fig. 4. (left) For the dynamic network in Fig. 1 the above DAG shows the transmission of selected plans p by high priority vehicles to lower priority vehicles - low number denote high priority. (right) Two vehicles that enter each other's comm range at maximum velocity, cannot collide if after finishing their plans they execute their contingency plans.

collision all the trajectories that intersect the contingencies of other vehicles to satisfy Eq. 4.

Req. 3 specifies that when a vehicle makes a decision, it must inform the other vehicles so that pairs of plans satisfy Eq. 5. These messages follow the vehicles' priorities. The highest priority vehicle V_i computes a solution plan $p_i(t_n : t_{n+1})$ from the motion planner and the accompanying contingency that could be executed at state $x^{\pi_i}(t_{n+1})$. V_i transmits its solution to lower priority vehicles, which must now come up with a plan that respects Eq. 5 given V_i's choice. Every vehicle waits to receive the choices of vehicles with higher priority before selecting a plan. If a plan that respects Eq. 5 is available in the tree structure, it is selected and transmitted to lower priority vehicles. If no plan is found, the available contingency is selected and transmitted. If time is running out (variable $PlanningCycle$ in Algorithm 1) and not all higher priority vehicles have send their plans, then a contingency is again selected.

Note that the prioritized scheme imposes a total ordering over all the vehicles. In the worst case, this may result in the lowest priority vehicle having to wait for all other vehicles to select plans. High priority vehicles have to transmit their selection early enough (variable $PlanningBudget$ in Algorithm 1) so that the sequence of selected plans reaches low priority vehicles within the planning cycle. *Even if the $PlanningBudget$ is not sufficiently long so that all vehicles have time to communicate, the vehicles still do not collide in our setup*. They will end up selecting contingencies and stop safely. Although this undesired effect is less pronounced when vehicles have limited communication, it is a disadvantage of the prioritized scheme. We have, however, addressed this issue by proposing a fully distributed approach as an extension of this work that guarantees the satisfaction of the three requirements without priorities [2].

C. Safe Coordinated Planning: Limited Communication

When vehicles have limited communication range, dynamic networks are formed and dissolved as the vehicles move towards their goals (Fig. 1). This, however, does not considerably effect the algorithm as long as two vehicles not within range cannot collide until they approach one another and communicate. Given enough space to deaccelerate and

come to a complete stop the collision can be avoided.

The Invariant can still be guaranteed by imposing limits on the maximum velocity of the vehicles by taking into account the worst case scenario, shown in Fig. 4 (right). Two vehicles are just outside range and they move with maximum velocity towards one another. Then they will keep approaching one another for an entire cycle with maximum velocity. At the end of the cycle, however, they will communicate. If they manage to find compatible plans, they will continue operating normally. Otherwise, they must execute contingencies. The Invariant can still be satisfied as long as the following is true: the distance that a vehicle covers until it comes to a complete stop when it moves at maximum velocity for one planning cycle and then applies a contingency plan must be less than half of the communication range. For some realistic parameters for car-like vehicles (comm. range 100m, breaking deacceleration 10m/sec^2, planning cycle $1sec$) the allowable maximum velocity is considerably high (approx. $80Km/h$ or $50mph$).

In the case of limited communication the flow of information is not a chain. The Directed Acyclic Graph (DAG) in Fig. 4 (left), shows the flow of information and the partial ordering defined by the priorities of the dynamic vehicular network displayed in Fig. 1(left). The DAG structure allows for the planning and selections steps to be executed in parallel on many vehicles even with a prioritized scheme.

D. Retaining Communication

It is also easy to satisfy the constraint that the vehicles maintain a communication network while moving. Assume the vehicles form a communication graph as in Fig. 1(left) and the objective is to move as a vehicular network. To satisfy the network constraint, we need the communication graph to remain connected. For the latter, it is sufficient to retain communication links along a spanning tree of the communication graph. There are efficient algorithms that can compute a spanning tree distributedly. This can be done in the beginning of every planning cycle. The planning algorithm has then to guarantee that the vehicles do not choose trajectories that will break the communication links along the spanning tree.

Assume V_i, V_j share an edge e_{ij} on the spanning tree. We can make sure that e_{ij} will not break if we treat as collision any pair of trajectories that concatenated with the corresponding contingencies bring V_i, V_j out of range. This amounts to just adding an extra check for requirements 2 and 3 for the pairs of vehicles that share edges of the spanning tree. Trajectories that break spanning tree edges, are not considered compatible. Since the vehicles move, the communication graph can change (Fig. 5 (right)). Consequently, the spanning tree recomputed in every cycle also changes over time. This allows the network to achieve different topology if it is required. Note that for an edge to be considered as a valid communication link, it must be retainable during a planning cycle given the dynamic motion constraints.

Car-Like

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{V} \\ \dot{s} \end{pmatrix} = \begin{pmatrix} \cos\theta \cdot \cos s \cdot V \\ \sin\theta \cdot \cos s \cdot V \\ \sin s \cdot V \\ \alpha \\ t \end{pmatrix}$$

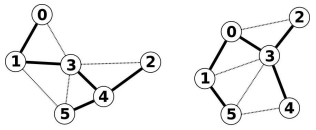

Fig. 5. The state update equations for the car-like vehicles(left). A communication graph and the spanning tree change as the vehicles move, but the vehicles remain a network (right).

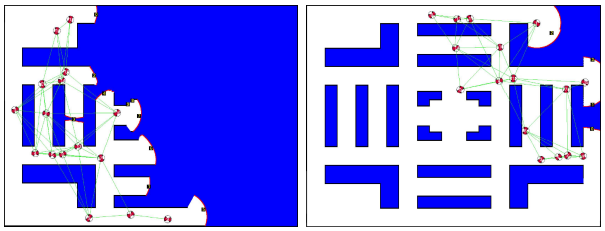

Fig. 6. Two snapshots of 16 vehicles exploring the labyrinth environment, while retaining a vehicular network.

IV. Distributed Simulation

As a proof of concept that the Alg. 1 can avoid collisions, we implemented it on a distributed vehicular simulator and experiments were ran on a Cray XD1 cluster. Each vehicle is simulated on a different processor and operates under time limitations so as to implement the replanning operation. The simulated vehicles exchange messages using sockets only if they can communicate. Vehicles have also limited sensing range. and car-like motion equations as shown in Fig. 5. Each vehicle has velocity bounds : $|V| \leq 3.5\ m/s$, acceleration bounds: $\alpha \leq 0.8\ m^2/s$ as well as steering bounds $|s| \leq 1\ deg/m$, $|t| \leq 4\ deg/(m \cdot s)$. Each vehicle has only one type of contingency plan, a breaking maneuver that brings it to a complete stop as soon as possible.

Algorithm 1 is tested on a scenario where a vehicular network explores an environment. This problem combines many challenges: unknown workspace, multiple vehicles with dynamics, network retainment etc. It also involves a variety of issues not discussed in this paper such as sensing, coordinating motions to maximize coverage, etc. Figs. 6 and 7 present two experiments where our technique is used to coordinate the motion of multiple vehicles that explore an unknown environment. The vehicles start at the bottom left corner in the scene, and the workspace is unknown. To promote exploration, the vehicles set as goals the frontiers of the unknown area. Experiments were executed in environments with a lot of narrow passages that force numerous encounters between vehicles.

Table I summarizes our main results in terms of safety. We consider teams of 2 to 16 vehicles that replan online with a planning cycle of 1.5 sec. We measure the time (in seconds), that the vehicles can move without colliding with each other when Reqs. 2 and/or 3 from section III-B are relaxed. The numbers reported show the time at which the first collision or loss of network connectivity occurs. The problem is so constrained for 16 robots that often collisions cannot be avoided past the 2nd replanning loop. The results

	Req 1		Req1 & Req2		Req1 & Req3		All Requirements	
Nr Vehicles	1^{st} failure (sec)	success %	1^{st} failure (sec)	success %	1^{st} failure(sec)	success %	1^{st} failure(sec)	success %
2	287.10	10%	293.25	37.37%	113.10	0%	N/A	100%
4	21.00	0%	141.07	12.00%	21.53	0%	N/A	100%
8	3.67	0%	24.16	0%	4.31	0%	N/A	100%
16	3.00	0%	23.10	0%	3.00	0%	N/A	100%

TABLE I

PROBABILITY THAT NETWORKS OF CAR-LIKE VEHICLES SUCCEED TO EXPLORE WHEN DIFFERENT REQUIREMENTS ARE MET.

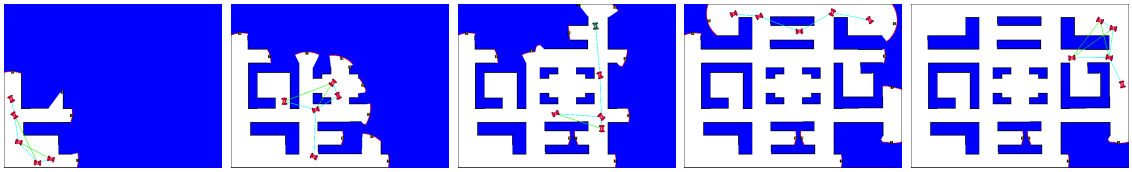

Fig. 7. Snapshots from an experiment in scene "labyrinth" with 5 vehicles, communication range at 25% of the scene width and sensing range at 15%.

Scene:	Labyrinth		
Vehicles	2, 4, 8	16	32
%	< 0.5 %	1.35 %	8.42 %

TABLE II

AV. PERCENTAGE OF CYCLES A VEHICLE EXECUTES CONTINGENCY.

are averaged over 10 runs and are shown in columns labeled *failure*. Our theoretical analysis is confirmed: if one of the two requirements is absent, the vehicles collide with each other. When all requirements are enabled, then there is no failure. The columns labeled *success*, measure the percentage of successful exploration of the whole space without collisions. For small teams of 2 or 4 vehicles, there are cases where the vehicles completed the task without one or both of the requirements. This is to be expected since the chances of an encounter are lower for small teams.

An important question is whether the vehicles end up in a deadlock situation. Although we have not observed deadlocks in our experiments, it is not easy to show that in general deadlocks are avoided. This question is also related to higher-lever decision making such as goal assignment.

V. CONCLUSION

This paper focuses on the safety issues that arise when multiple vehicles with kinodynamic constraints operate in the same area. We extend techniques that avoid `ICS` to the case of multiple vehicles with limited communication. A decentralized prioritized algorithm that provably achieves safety is described and has been implemented on a distributed simulator. The experiments confirm our theoretical expectations. The framework allows for plugging in other types of kinodynamic constraints and the use of more elaborate decentralized schemes. We have extended it so that instead of priorities, a distributed message-passing protocol is employed that satisfies the theoretical requirements for safe motion coordination and has better scalability properties [2]. The issues of uncertainty and communication reliability are also of great importance and we intend to address them in the context of the proposed framework in future work.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their detailed and helpful comments.

REFERENCES

[1] K. E. Bekris and L. E. Kavraki. Greedy but safe replanning under kinodynamic constraints. In *ICRA*, Rome, Italy, April 2007.
[2] K. E. Bekris, K. Tsianos, and L. E. Kavraki. A distributed protocol for safe real-time planning of communicating vehicles with second-order dynamics. In *ROBOCOMM*, Athens, Grrece, Oct. 15-17 2007.
[3] M. Bennewitz, W. Burgard, and S. Thrun. Finding and optimizing solvable priority schemes for decoupled path planning for teams of mobile robots. *Robotics and Autonomous Systems*, 41(2):89–99, 2002.
[4] J. van den Berg, D. Ferguson, and J. Kuffner. Anytime path planning and replanning in dynamic environments. In *ICRA*, May 2006.
[5] J. Bruce and M. Veloso. Safe multi-robot navigation within dynamic constraints. *Proc. of the IEEE*, 2006.
[6] C. Clarc, S. Rock, and J.-C. Latombe. Dynamic networks for motion planning in multi-robot space systems. In *Intl. Symp. of Artificial Intelligence, Robotics and Automation in Space*, 2003.
[7] M. Christopher Clark, Tim Bretl, and Stephen Rock. Applying kinodynamic randomized motion planning with a dynamic priority system to multi-robot space systems. In *Aerospace Conference*, 2002.
[8] D. Ferguson, N. Kalra, and A. Stentz. Replanning with rrts. In *ICRA*, pages 1243–1248, Orlando, FL, May 15-19 2006.
[9] T. Fraichard and H. Asama. Inevitable collision states - a step towards safer robots? *Advanced Robotics*, 18(10):1001–1024, 2004.
[10] R. Gayle, K. R. Klinger, and P. G. Xavier. Lazy reconfiguration forest: An approach for planning with multiple tasks in dynamic environments. In *ICRA*, pages 1316–1323, Rome, April 10-14 2007.
[11] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock. Randomized kinodynamic planning with moving obstacles. *IJRR*, 21:233–255, '02.
[12] M. Kallman and M. Mataric. Motion planning using dynamic roadmaps. In *ICRA-04*, volume 5, pages 4399–4404, 2004.
[13] A. M. Ladd and L. E. Kavraki. Fast tree-based exploration of state space for robots with dynamics. In *WAFR*, pages 297–312, 2005.
[14] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. *IJRR*, 20(5):378–400, May 2001.
[15] S. Petti and T. Fraichard. Partial motion planning framework for reactive planning within dynamic environments. In *ICRA*, Barcelona, Spain, September 2005.
[16] M. Saha and P. Isto. Multi-robot motion planning by incremental coordination. In *IROS*, pages 5960–5963, Beijing, China, Oct. 2006.
[17] G. Sanchez and J.-C. Latombe. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. In *ISRR*, pages 404–417, 2003.
[18] X. Yang, L. Liu, N. H. Vaidya, and F. Zhao. A vehicle-to-vehicle communication protocol for cooperative collision warning. In *MOBIQUITOUS-04*, 22-26 Aug. 2004.
[19] M. Zucker, J. Kuffner, and M. Branicky. Multipartite rrts for rapid replanning in dynamic environments. In *ICRA-07*, pages 1603 – 1609, Rome, Italy, April 10-14 2007.