A Distributed Protocol for Safe Real-Time Planning of Communicating Vehicles with Second-Order Dynamics

Kostas E. Bekris Konstantinos I. Tsianos Lydia E. Kavraki
Computer Science Department, Rice University, Houston, TX, 77005
Email: {bekris, konstantinos, kavraki}@cs.rice.edu

***Abstract*—This work deals with the problem of planning in real-time, collision-free motions for multiple communicating vehicles that operate in the same, partially-observable environment. A challenging aspect of this problem is how to utilize communication so that vehicles do not reach states from which collisions cannot be avoided due to second-order motion constraints. This paper provides a distributed communication protocol for real-time planning that guarantees collision avoidance with obstacles and between vehicles. It can also allow the retainment of a communication network when the vehicles operate as a networked team. The algorithm is a novel integration of sampling-based motion planners with message-passing protocols for distributed constraint optimization. Each vehicle uses the motion planner to generate candidate feasible trajectories and the message-passing protocol for selecting a safe and compatible trajectory. The existence of such trajectories is guaranteed by the overall approach. Experiments on a distributed simulator built on a cluster of processors confirm the safety properties of the approach in applications such as coordinated exploration. Furthermore, the distributed protocol has better scalability properties when compared against typical priority-based schemes.**

I. Introduction

Autonomous vehicles have long been the focus of robotics research. The progress in wireless networking allows to consider groups of vehicles that operate in the same environment and use communication to coordinate their motion. Moreover, it gives rise to the idea of networks of vehicles that jointly solve a task while retaining connectivity. The control of such systems involves multiple research challenges. Here, we focus on motion planning issues. Given procedures for updating a vehicle's map, state and goal, the objective is to design feasible, collision-free trajectories for the vehicles.

We are interested in a solution with the following characteristics: (i) A general and abstract algorithm that is not limited to specific system dynamics or to specific types of workspaces and obstacles. (ii) A scalable, distributed solution that respects the physical limitations in sensing and communication and avoids centralized computation. (iii) A real-time algorithm, since vehicles do not typically have global knowledge of their workspace. This means that sensing, planning and execution are interleaved and there is limited amount of time to compute a partial plan towards the goal. (iv) A safe solution for systems with second-order constraints. The algorithm must provide guarantees for collision-avoidance and the retainment of a communication network if desired by the team.

A. Related Literature

Multiple techniques exist for decentralized motion planning [1]. In formation control agents move while maintaining preassigned relative positions, which can be achieved with potential-fields [2], [3], [4], leader-follower approaches [5], [6] or local control laws [7]. Decentralized, navigation functions [8], [9] provide a feedback solution and can be used for vehicles with independent goals. Most of these methods focus on providing elegant stability proofs. Despite their elegance, it is difficult to apply them in general state spaces (e.g. complex obstacle and robot shapes and dynamics) [10].

This paper investigates an alternative which is less dependent on the system's dynamics or the obstacle types. It utilizes sampling-based [11], kinodynamic planning [12] popularized by algorithms such as `RRT` [13], [14], [15]. Instead of constructing control laws given representations of the state space obstacles, such algorithms execute a search in the state space, which is mostly a computational rather than an analytical challenge. Their drawback is that they have weaker completeness properties and optimality guarantees are abandoned in favor of practical performance and generality.

The original sampling-based planners were offline methods and assumed known workspaces. A way to deal with partial-observability is to replan online and construct partial plans towards the goal given time limitations [16], [17]. When replanning with a sampling-based planner for a system with second-order dynamics, safety issues arise: a collision-free but partial plan may lead a vehicle to a state from which collisions cannot be avoided due to the dynamics (Inevitable Collision States (`ICS`) [18], [15]). This problem is particularly acute when multiple second-order vehicles operate in close proximity in the same environment. Similarly, a partial plan could also lead to states from which network connectivity will be inevitably lost. A framework that deals with `ICS` and real-time planning for a single vehicle has been recently developed in the sampling-based planning literature [18], [15].

The use of sampling-based planners on multi-agent problems is limited and it typically follows a centralized approach even for networks of vehicles [19]. Decoupled approaches, which are incomplete but more efficient, can utilize sampling-based planners [20]. In previous work [21], a priority-based scheme was employed where each agent employs plans given the paths computed by higher-priority agents. Instead of priorities, this paper studies message-passing algorithms for coordination related to loopy belief propagation, a method for distributed optimization in constraint networks [22], [23]. These message passing algorithms have been successfully applied to solve distributed inference problems in wireless sensor networks [24].

B. Contribution

This paper describes a novel integration of sampling-based kinodynamic planners [10], [14] with message-passing protocols [23], [24] to distributedly control the motion of multiple communicating vehicles. It is an extension of work on safe, real-time sampling-based planning [15] to the case of multiple networked vehicles with limited communication range. It can guarantee safety in terms of avoiding inevitable collision or loss of connectivity states. Compared to alternative approaches for decentralized motion planning [7], [8] it is easily implementable on general workspaces and to systems with different dynamics. In contrast to existing work on motion planning for dynamic networks, where coordination is centralized [19], the approach is distributed.

The starting point for the method is the identification of the information that must be exchanged between the vehicles so as to plan safe trajectories. This information requirements dictate our communication protocol. Within the protocol, each vehicle uses a sampling-based planner [15] to generate feasible trajectories that allow the existence of safe alternatives to other vehicles. Then the vehicles coordinated through message passing [23], [24] to select their trajectory. The existence of safe, compatible solutions is guaranteed by the algorithm. Among the safe solutions and given the available time, the asynchronous protocol optimizes a joint payoff function.

The proposed method has been implemented on a multiprocessor simulator. Each processor models a vehicle and communicates asynchronously with other processors. The experimental results confirm the theoretical guarantees of collision avoidance and network retainment for second-order vehicles jointly exploring an unknown workspace. The distributed protocol has computational advantages when compared against prioritized schemes [21].

II. Problem Setup

Consider vehicles $V = \{V_1, \ldots, V_v\}$ operating in a world with obstacles. Both obstacles and vehicles are rigid-bodies and there are no restrictions on their shape. Each vehicle is able to sense a local region around it and can communicate with other vehicles within a limited range. Each vehicle V_i is a dynamic system whose motion is governed by differential equations of the form:

$$\dot{x_i}(t) = f(x_i(t), u_i), \qquad g(x_i(t), \dot{x_i}(t)) \leq 0 \tag{1}$$

where $x_i(t) \in \mathcal{X}_i$ represents a state, $u_i \in \mathcal{U}_u$ is a control, f, g are smooth and t is time. This paper focuses on systems with bounds both in velocity and acceleration. The dynamics of the systems we experimented with can be found in Section IV.

Given the communication limitations and states $\{x_1(t), \ldots, x_v(t)\}$, the vehicles form dynamic communication links represented by a graph $G(t) = \{V(t), E(t)\}$, where $e_{ij} \in E(t)$ as long as V_i, V_j are within range. The neighbors of V_i in the graph $G(t)$ are denoted as $N_i(t)$.

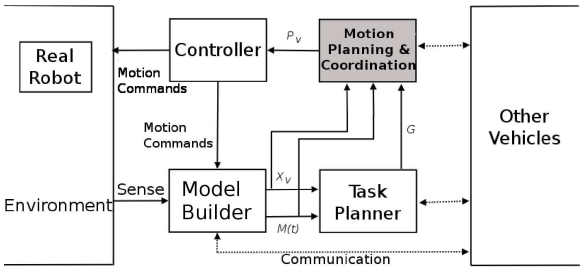

Fig. 1. The closed loop architecture and modules on a single vehicle.

A. High-Level Replanning Framework

The vehicles execute tasks which require motion. While moving, the vehicles must avoid collisions both with obstacles and with other vehicles. Since the workspace is only partially observable, the vehicles must update their world model and state estimate given new sensory information. In parallel, they must replan in real-time trajectories towards their goals. To achieve this objective, a vehicle's function is broken down into a sequence of consecutive operational cycles. The various vehicle operations, shown in Fig. 1, are executed in a pipeline over these cycles. For cycle $(t : t + dt)$ vehicle V_i executes the following:

1. Up to time t: a localization and mapping routine updates the map $M_i(t)$ and estimates future state $x_i(t + dt)$.
2. Given the map, a goal $G_i(t)$ is computed for V_i.
3. Given $M_i(t)$, $G_i(t)$ a planner must compute a plan $p(dt)$ before $t + dt$.
4. At time $t + dt$ the plan $p(dt)$ is executed at $x_i(t + dt)$.

In this work we focus on step 3, i.e. on how to utilize communication so as to provide safety guarantees when multiple vehicles operate in close proximity. Each vehicle can only communicate with neighboring vehicles given the communication constraints and exchange information. We will specify what kind of information has to be exchanged to guarantee collision avoidance. In this work, we do not deal with issues related to uncertainty in sensing and action as well as unreliable communication.

B. Motion Planning Notation

The objective of the motion planner is to compute a **plan** $p(dt)$, which is a time sequence of controls: $p(dt) = \{(u_1, dt_1), \ldots, (u_n, dt_n)\}$, where $dt = \sum_i dt_i$. When a plan $p(dt)$ is executed at state $x(t)$, a vehicle will follow the **trajectory**: $\pi(x(t), p(dt))$. A trajectory is feasible, if it respects the constraint functions f and g from Eq. 1.

A state along trajectory $\pi(x(t), p(dt))$ at time $t' \in [t : t+dt]$ is denoted as $x^{\pi}(\ t'\ )$. When a vehicle executes a plan $p(dt)$ from state $x(t)$ and consecutively executes plan $p'(dt')$, then the resulting **trajectory concatenation** will be denoted as:

$$\pi'(\ \pi(x(t), p(dt)),\ p'(dt')\ ).$$

If two vehicles V_i, V_j at time t are not in collision with each other or with obstacles, then their corresponding states $x_i(t), x_j(t)$ are **compatible states**: $x_i(t) \asymp x_j(t)$.

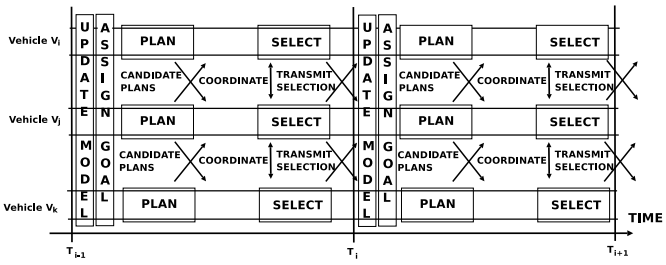

Fig. 2. The operation that a single vehicle executes in two consecutive planning cycles.

Two 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** ($\pi_i \asymp \pi_j$) if the two trajectories do not cause collisions with workspace obstacles and:

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

Compatible trajectories between vehicles may still lead to an inevitable collision state from which a collision cannot be avoided in the future due to second-order constraints [18]. A state $x_i(t)$ is an **Inevitable Collision State (ICS)** given the states $\{x_1(t), \ldots, x_v(t)\}$ if $\forall\ \pi_i(x_i(t), p_i(\infty))$:

$$\exists\ (dt \wedge j \neq i) \text{ so that } \forall\ \pi_j(x_j(t), p_j(\infty)) \\ \text{states } x^{\pi_i}(dt) \text{ and } x^{\pi_j}(dt) \text{ are not compatible.}$$

C. Problem Definition

Given the map $M_i(t)$ and a state estimate $x_i(t + dt)$, the motion planning module of each vehicle V_i must compute before time $(t+dt)$ a plan $p_i(dt')$ so that given the trajectories of all other vehicles $\pi_j(x_j(t + dt), p_j(dt'))$ $(\forall j \neq i)$:

- $\pi_i(x_i(t + dt), p_i(dt')) \asymp \pi_j(x_j(t + dt), p_j(dt'))$
- State $x_i^{\pi_i}(dt')$ is not ICS.

Vehicle V_i can only communicate with vehicles in the set $N_i(t)$. A secondary objective for the planner is to refine the quality of the selected trajectory given a measure of path quality and a goal $G_i(t)$.

III. Distributed Protocol for Safe, Real-Time Motion Planning

There are two basic implementation choices in the proposed approach. Motion coordination is achieved in a decoupled manner. This feature distinguishes our approach when compared against related work on planning for communicating vehicles [19], where the vehicles forming a temporary dynamic network solve a centralized problem. Moreover, the motion planning and coordination operation of each vehicle are split into two separate steps as shown in Fig. 2:

1) **Generate Candidate Plans:** During the first step, the algorithm searches the state space of vehicle V_i so as to generate a set of candidate plans $\mathcal{P}_i$.
2) **Select Compatible Plans:** During the second step, neighboring vehicles communicate by exchanging sets $\mathcal{P}_i$ and evaluating their performance in terms of collision avoidance and task execution.

For the plan generation step, sampling-based, kinodynamic planners are particularly appropriate to search the state space and produce multiple candidate valid plans $\mathcal{P}_i$ that are at least collision-free with the workspace obstacles.

For the plan selection step, the exchanged plans $\mathcal{P}_i$ are viewed as actions in a discrete action space. Two plans of different vehicles are acceptable solutions if the corresponding trajectories are compatible. Then the problem of distributedly selecting compatible trajectories is reduced to a distributed constraint optimization problem, for which message-passing algorithms exist [23], [24].

A. Plan Generation: Sampling-based Planning

There are various alternatives on how to implement a sampling-based, kinodynamic planner [13], [14], [15]. In Algorithm 1 we present an abstract version of such algorithms. The algorithm constructs a tree data structure (variable $Tree$), where nodes of the tree correspond to states and edges correspond to trajectories. The tree is rooted at the state $x(t + dt)$ and sampling is used to propagate feasible trajectories into the state space. The algorithms as described here searches the state space of a single vehicle given only workspace obstacles and ignores the effects of other vehicles. We will later show what changes must be made so that collisions with other vehicles are also avoided.

Algorithm 1 SAMPLING-BASED PLANNER

$Tree \leftarrow$ Retain valid subset of $Tree$ from previous cycle
while $(time < PlanningBudget)$ **do**
{
  Select a state $x(t')$ on the existing $Tree$
  Select valid plan $p(dt')$ given state $x(t')$
  Forward propagate trajectory $\pi(x(t'), p(dt')$
  if (π is not **collision-free** with obstacles) **then**
    Reject π
  else
    Add π to $Tree$
}

In our implementation we have followed an approach where the expansion of the tree is biased so as to greedily expand the data structure towards the goal while still providing the weak completeness guarantees of sampling-based planning [15]. Since the algorithm is executed in a continuous replanning loop, part of the tree data structure from the previous planning cycle may still be valid in the beginning of a new cycle. The valid part of the tree is retained as in many recent replanning approaches [16], [17] to accelerate performance and improve path quality. The planner is executed for a limited amount of time (variable $PlanningBudget$, which should be less than the total cycle duration dt) and then it is interrupted. At that point we can extract from $Tree$ all the trajectories that are rooted at state $x(t + dt)$ and which have duration dt. These trajectories are by construction feasible and collision-free with workspace obstacles. We store these trajectories as candidate plans $\mathcal{P}_i$, for vehicle V_i.

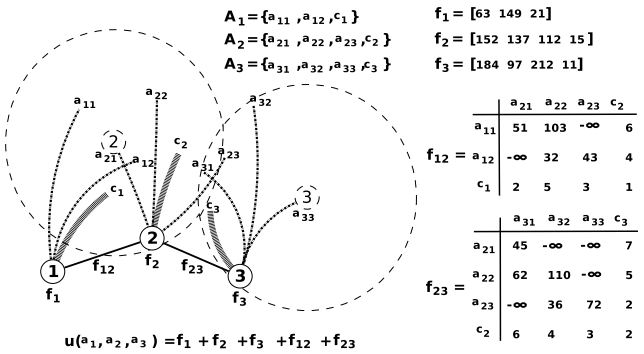

Fig. 3. A simple coordination graph, the action sets A_1, A_2, A_3, the atomic and pairwise payoffs $f_1, f_2, f_3, f_{12}, f_{23}$ and the global utility function u

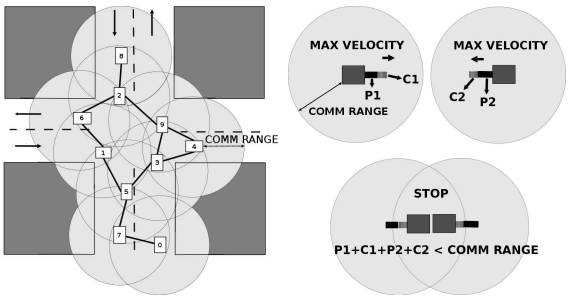

Fig. 4. (left)A coordination graph for 10 vehicles. (right) Safety assumption

B. Plan Selection: Message-Passing Protocol

After the plan generation step, each agent has a discrete set of actions to select from. The objective of the path selection step is to distributedly assign trajectories to vehicles so that there is no incompatible pair of trajectories and as a secondary goal to select good quality trajectories. Such problems can be modeled with coordination graphs [25] and can be solved with distributed message passing algorithms based on *belief propagation* [22], such as the *max-plus* algorithm [23].

In this formalization, we assume we have n agents, and each agent i has to select an action a_i out of a finite action set A_i. Generally the goal is to find the optimal action vector $\overline{a}^* = (a_1^*, \ldots, a_n^*)$ that maximizes a global utility function $u(\overline{a})$. The utility has a structure captured by a coordination graph $CG = (V, E)$. On every node of CG we define a function $f_i(a_i)$ called *atomic payoff*. An atomic payoff describes how well each action serves the goal of the agent corresponding to that node. On the edges e_{ij} of CG we define *pairwise payoff* functions $f_{ij}(a_i, a_j)$ that indicate how good for the team are pairs of actions of interacting agents (see Fig. 3 for an example). The global utility is assumed to depend only on the unary and pairwise payoff functions as follows:

$$u(\overline{a}) = \sum_i f_i(a_i) + \sum_{e_{ij} \in E(CG)} f_{ij}(a_i, a_j)$$

Max-plus is a distributed message passing algorithm that attempts to compute an optimal action vector using only local computations and communication for every agent. While the algorithm is running, each agent i chooses a neighboring agent j on CG, collects and adds all incoming messages from other agents in its neighborhood, and sends a new message to j that is computed by the following formula:

$$m_{ij}(a_j) = \max_{a_i} \{ f_i(a_i) + f_{ij}(a_i, a_j) + \sum_{k \in N(i), k \neq j} m_{ki} \} \quad (2)$$

At any time during the execution, the agents can compute a *marginal* function $g_i(a_i) = f_i(a_i) + \sum_{k \in N(i)} m_{ki}$. Maximizing g_i provides the best possible action a_i' for agent i with respect to messages from other agents. $u(a_1', \ldots, a_n')$ is an approximation to the optimal u.

In order to adapt this formulation in our framework, each vehicle can be viewed as a node in the coordination graph CG (see Fig. 4(left)). Two nodes share an edge in the graph if the corresponding vehicles can communicate or if they can potentially collide. We want to avoid, however, the case where two vehicles can end up in inevitable collision states before they have time to communicate. That is why we impose a maximum velocity limit that depends on the duration of the replanning cycle and the communication range. This limit must guarantee that two non-communicating vehicles which move with maximum speed one towards the other will have enough time to communicate, select contingenecy plans and stop before colliding. This scenario is described in Fig. 4. Given the velocity limit, the edges of CG correspond to pairs of vehicles that can communicate during the entire cycle.

The discrete set of actions of the max-plus algorithm corresponds to the set of candidate plans $\mathcal{P}_i$. The atomic payoffs $f_i(p_i)$, where $p_i \in \mathcal{P}_i$ can be computed by evaluating how close each trajectory takes vehicle V_i to its goal G_i. In the evaluation of the pairwise payoffs we must also express whether two trajectories are compatible or not:

$$f_i(p_i(dt), p_j(dt)) = -\infty \text{ if}$$
$$\pi_i(x_i(t+dt), p_i(dt)) \not\succ \pi_j(x_j(t+dt), p_j(dt)) \quad (3)$$

where $p_i \in \mathcal{P}_i$ and $p_j \in \mathcal{P}_j$. If the two plans p_i and p_j are compatible then the pairwise payoff can be assigned a positive value that depends on other properties that are required to be optimized. Consequently, it is necessary before the coordination step for the neighboring vehicles to exchange the sets of candidate plans so as to compute the pairwise payoff functions.

The pairwise payoffs can be computed in a distributed way that balances the burden among vehicles. Each vehicle computes one row for every pairwise payoff matrix that it is involved in, in a cyclic order. Then each vehicle transmits this row to its neighbors. In the computation of f_{ij} if $i \leq j$ then i computes rows $r_1, r_2, \ldots$ and j computes in reverse $r_{max}, r_{max-1}, \ldots$, until the whole array is completed. In this way, a vehicle that has many neighbors is not overwhelmed and its computational overhead is outsourced to neighbors with a smaller number of payoff matrices to compute.

C. Achieving Safety

The two step approach we have described so far is a distributed search technique, which has no guarantee of finding compatible plans. It can actually fail for multiple reasons:

1) Imposing time limitations on sampling-based planners may result in $\mathcal{P}_i$ being empty even if there is a solution.
2) The cycle may be completed before vehicles have enough time to exchange and evaluate plans.
3) Compatible trajectories may exist but they may not have been produced because decoupled coordination is incomplete [20].
4) Max-plus is also incomplete when CG is not a tree. Although it converges quickly to a near-optimal solution [23], incompatible pairs may still be selected.
5) The approach so far does not address the issue of ICS.

All these safety concerns can be dealt with, if every vehichle has available in the beginning of every planning cycle a contingency plan $\gamma(\infty)$ that avoids collisions with the environment and other obstacles. For example, for systems with second order dynamics and static workspace obstacles, a contingency plan can be a breaking maneuver that brings a vehicle to a complete stop. In [21] a theoretical study of avoiding ICS in the case of multiple communicating vehicles by using contingency plans can be found. That paper describes the following sufficient requirements so as to guarantee the existence of a contingency plan in the beginning of every replanning cycle:

Requirement 1: The concatenation of a plan $p_i(dt)$ with a contingency plan $\gamma_i(\infty)$ must be collision-free with respect to workspace obstacles:

$$\pi'(\ \pi(x(t_n), p_i(dt)),\ \gamma_i(\infty)\ ) \text{ is collision-free.} \quad (4)$$

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

$$\begin{aligned} \forall j \neq i: \quad & \pi_i'(\ \pi_i(x_i(t_n), p_i(dt)), \gamma_i(\infty)) \\ & \asymp\ \pi_j(x_j(t_n), \gamma_j(\infty)) \end{aligned} \quad (5)$$

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

$$\begin{aligned} \forall j \neq i: \quad & \pi_i'(\ \pi_i(x_i(t_n), p_i(dt)), \gamma_i(\infty)) \\ & \asymp \pi_j'(\ \pi_j(x_j(t_n), p_j(dt)), \gamma_j(\infty)) \end{aligned} \quad (6)$$

These requirements are directives that decribe what kind of information needs to be exchanged between vehicles, and how the selection of future plans must be done. The first two requirements make sure that the planner will produce plans that respect the other vehicle's contingencies. The third requirements ensures that the contingency plans attached to selected plans (as in Eq. 6) are also compatible. In [21] an inductive proof is provided that shows that if the requirements are satisfied then the vehicles will be moving safely. In that work the coordination step is achieved with a priority-based scheme, while in the current work we show a fully distributed approach that does not use priorities. These requirements, however, do not depend on the planning algorithm or the coordination scheme used. In this work we have extended the previous contribution [21] by showing that the full scheme does not depend on a prioritized scheme for coordination, but that it can be also implemented with a fully distributed message passing protocol.

D. Algorithm

We describe here how we the overall proposed protocol can satisfy the requirements in a distributed way. The algorithm is provided in pseudocode in Algorithm 1.

To satisfy the second requirement, each vehicle V_i must be aware of the contingency plans of neighboring vehicles V_j at state $x(t + dt)$. These contingency plans have already been computed by each V_j during the previous step. This information can be communicated at the beginning of each cycle between neighbors. After exchanging contingency plans, the sampling-based, kinodynamic planner is invoked. The planner now not only produces collision-free trajectories for duration dt but also avoids ICS with obstacles (Eq. 4). The planner considers as colliding all the trajectories that intersect the contingencies of other vehicles to satisfy Eq. 5.

Next, the set of candidate plans $\mathcal{P}_i$ is constructed from $Tree$. For each plan in this set, the corresponding contingency is attached to it and the unary payoff $f_i(p_i)$ is evaluated. Then, neighboring vehicles exchange their candidate plans and compute pairwise payoffs. Instead of using Eq. 3, however, for defining incompatibility, we must use Eq. 6, which takes the concatenation with contingency plans into account. Given the definition of unary and pairwise payoffs the asynchronous message-passing protocol is initiated and the vehicles start exchanging messages. When the algorithm runs out of time, vehicle transmit to their neighbors their final action selections. Max-plus is incomplete, so if two neighboring vehicles have selected incompatible trajectories then one of the two vehicles switches to the contingency plan. This is a very fast adjustment step, which guarantees that the third requirement is always satisfied. In the experiments section we show that the cases where max-plus has to resort to the contingency plan are fewer compared to priority-based schemes.

The secondary goal is to find among the safe solutions one that maximizes the global utility u, within the allocated amount of time. Because the algorithm does not monotonically increase the global utility, it must periodically compute u and keep track of the action vector that produced the maximum value. However, no single agent has all the available information to compute u. In order to achieve an efficient, distributed computation of the utility we use a minimum spanning tree of CG. There are asynchronous distributed protocols for computing minimum spanning trees on graphs using local information such as the distance between agents [26]. Given the minimum spanning tree structure, an arbitrary vehicle acting as a root of the tree initiates the process:

Algorithm 2 DISTRIBUTED, SAFE PLANNER

Identify set of neighbors N_i
`(Exchange contingencies)`
for all $j \in N_i$ **do**
  Send contingency $\gamma_i(\infty)$ to V_j
  Receive contingency $\gamma_j(\infty)$ from V_j
`(Planning: respects Req. 1,2)`
$Tree \leftarrow$ Retain valid subset of $Tree$ from previous cycle
while $(time < PlanningBudget)$ **do**
  Select a state $x(t')$ on the existing $Tree$
  Select valid plan $p(dt')$
  Propagate trajectory $\pi(x(t), p(dt')$
  if ($(\pi(\pi(x(t), p(dt'), \gamma(\infty))))$ is not **collision-free**) **then**
    Reject π
  else
    for all $j \in N_i$ and while π not rejected **do**
      if ($\pi(x(t), p(dt')) \not\asymp \pi_j(x_j(t_n), \gamma_j(\infty))$) **then**
        Reject π
`(Evaluate and exchange candidate plans)`
$\mathcal{P}_i \leftarrow$ plans of duration dt from $Tree$
for all $p_i \in \mathcal{P}_i$ **do**
  Attach contingency plan $\gamma_i(\infty)$ to $p_i(dt)$
  Evaluate unary payoff $f_i(p_i)$ for $p_i \in \mathcal{P}_i$ given G_i
for all $j \in N_i$ **do**
  Send set $\mathcal{P}_i$ to V_j
  Receive set $\mathcal{P}_j$ from V_j
  `(Take Req. 3 into account)`
  for all $p_i \in \mathcal{P}_i$ **do**
    for all $p_j \in \mathcal{P}_j$ **do**
      if $(\pi_i'(\pi_i(x_i(t_n), p_i(dt)), \gamma_i(\infty)) \asymp \pi_j'(\ \pi_j(x_j(t_n), p_j(dt)), \gamma_j(\infty))$ **then**
        Compute payoff $f_{ij}(p_i, p_j)$ given the goals G_i, G_j
      else
        $f_{ij}(p_i, p_j) = -\infty$
`(Coordination)`
Enter into asynchronous message-passing to optimize:
$u(\overline{p}) = \sum_i f_i(p_i) + \sum_{e_{ij} \in E(CG)} f_{ij}(p_i, p_j)$
Stop protocol before time $t + dt$
Select plan $\overline{p_i}$ that maximized $u(\overline{p})$
for all $j \in N_i$ **do**
  if ($\overline{p_i}$ incompatible with $\overline{p_j}$) **then**
    Select contingency γ_i as the next action

Down pass The root transmits a signal to compute u. Each node passes the signal down the tree.

Up pass Each node i:

1) Collects partial payoff values and actions from children.
2) Maximizes marginal g_i and chooses best action a_i.
3) Adds its contribution to the global payofff u.
4) Sends new partial payoff and actions to its parent.

Down pass The root adds up all partial payoffs so as to compute and maximize u. The optimal value of u^* and actions $\overline{a}^*$ are transmitted down the tree.

This computation is fast since the utility computation messages can be interleaved with normal max-plus messages [23].

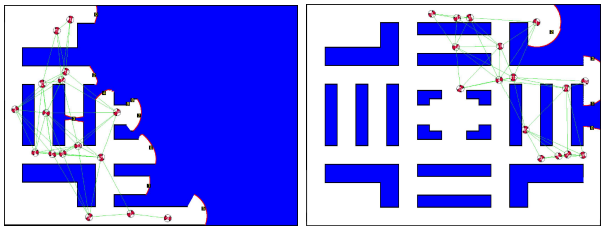

Fig. 5. Two snapshots of 16 vehicles exploring the labyrinth environment, while retaining a vehicular network.

E. Extension to Vehicular Networks

So far we have assumed that the vehicles execute tasks which do not require from the vehicles to retain a communication network. However, the same framework can be used to produce trajectories that also provide safety guarantees in terms of network connectivity. This can be easily achieved by considering as incompatible every pair of trajectories that breaks a communication link. In order to take the second-order dynamic constraints into account, we must also check that the concatenation of the trajectories with contingency plans also retains the communication link similarly to Eq. 6.

Yet, trying to maintain all the communication links is overly constraining. It is enough to try and maintain the edges in a minimum spanning tree of the communication graph. The minimum spanning tree structure can be computed distributedly in the beginning of each cycle [26]. In this case, only communication links along the minimum spanning tree are required to remain connected. If only a spanning tree of the communication graph is retained during each step, then the network is able to change the topology of its communication graph so as to adapt to its workspace.

IV. Experimental Results

Setup: We tested our algorithm on a distributed simulator that we developed and ran on an XD1 Cray cluster. The planner for each vehicle is running on a different processor and operates under time limitations imposed by a server that simulates ground truth. All data exchange is done via simple send and receive messages using sockets.

The simulated vehicular networks have been tested in three different environments. *Rooms* and *Random* are seen in Fig. 6. The first represents a structured environment with rooms and corridors, while the second is an unstructured environment. *Labyrinth* (Fig. 5) is a difficult scene that contains multiple narrow passages. Two types of vehicles have been tested, car-like robots, for which the dynamic equations are shown in

Car-Like

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

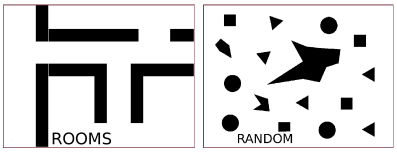

Fig. 6. The state update equations for the car-like vehicles and scenes "rooms" and "random".

	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. 6, and differential drive robots with bounded acceleration. The car-like robots obey velocity bounds : $|V| \leq 3.5m/s$, and acceleration bounds: $\alpha \leq 0.8m^2/s$ as well as steering bounds: $|s| \leq 1deg/m$, $|t| \leq 4deg/s$. Vehicles have limited sensing and communication ranges. For contingencies, deacceleration maneuvers were used. The framework allows for plugging in other types of dynamics and having different contingencies.

We present here results from an application that combines many of the constraints we are interested in testing our algorithm. The vehicles have to solve a coordinated exploration task while retaining a network and avoiding collisions. They are initially located at the bottom left corner of the environment, close one to another, but at collision-free states forming a network with a single component. During each replanning cycle a simulated model builder and a task planner transmit to the vehicles the updated map and set of goals. The goals correspond to frontiers of the unexplored space and are assigned greedily so that large frontiers which are close to vehicles are being considered first. Experiments with up to 32 vehicles have been conducted.

We compare the max-plus algorithm against a simpler prioritized scheme described in more detail in [21]. In that scheme, the vehicles have unique global priorities. The planning step is the same as here. For the plan selection, each vehicle receives the choices of higher priority vehicles and then tries to choose its own plan so that it is compatible the higher priority neighbors. If no such plan exists, the vehicle chooses the contingency plan. At last the vehicle transmit its selection to its lower priority neighbors.

Feasibility: Table I exhibits the importance of the safety requirements in decoupled replanning. We measure the time (in seconds), that the vehicles can move without colliding with each other when Requirements 2 and/or 3 (those necessary for safe multi-vehicle planning) 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 multiple vehicles, that often collisions cannot be avoided past the 2nd replanning loop. The results are averaged out of 10 runs and are shown in columns labeled *failure*. If either one of the two requirements is absent, the vehicles cannot avoid collisions with each other. When all requirements are enabled, then as expected, there is no failure. The columns labeled *success*, measure the percentage of successful exploration of the whole space without collisions. As we see, for small teams of 2 or 4 vehicles, there were some 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 such small teams.

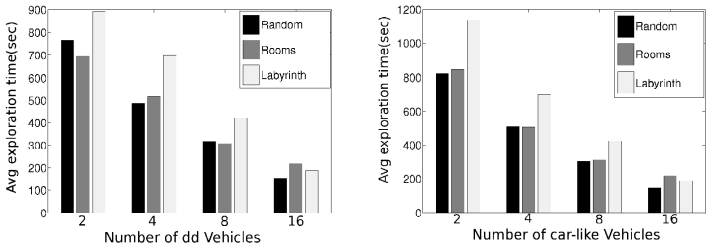

Fig. 8. Scalability results for three scenes: Random, Rooms, Labyrinth. Left: DD robots, Right: Car-Like robot

Contingency Plans: One important advantage of the max-plus algorithm is that it avoids the use of priorities in coordination. In priority-based schemes, lower priority vehicles are overly constrainted by the choices of higher priority agents. This may lead to frequent selection of contingency plans. We have experimented in scenes *Labyrinth* and *Rooms* for networks with 16 and 32 vehicles. Table II presents the number of times contingencies were selected using the simple prioritized scheme and max-plus. For 32 vehicles, max-plus chooses contingency plans considerably fewer times. Additionally, the results from the prioritized scheme have higher variation between different scenes and team sizes, while max-plus is much more consistent.

Scalability: The scalability properties of the algorithm are presented in Fig. 8, which provides the average running times (10 runs per case) to complete exploration in the three scenes for car-like and DD vehicles. Increasing teams size from 2 to 16 results in 5 to 6 times faster exploration. This is a very encouraging result given that the simulated systems are very constrained, both due to network and kinodynamic constraints. Moreover, there is no significant variation in the performance of the algorithm when applied to systems with different dynamics.

Performance and Parameter Dependence: Fig. 7(left) shows the average activity profile of a vehicle during each cycle. The algorithm utilizes most of the replanning cycle in useful computations but the payoff computation takes up a non-trivial amount of time. In the selection step, max-plus does not let the processor idle, and in most cases is able to find an optimal or near optimal solution. The latter is confirmed by

	Rooms		Labyrinth	
	16	32	16	32
Prioritized	3.61 %	24.5 %	1.35 %	8.42 %
Max-plus	0.98 %	2.26 %	3.04 %	4.84 %

TABLE II

AVERAGE PERCENTAGE OF CYCLES THAT V_i EXECUTES CONTINGENCY.

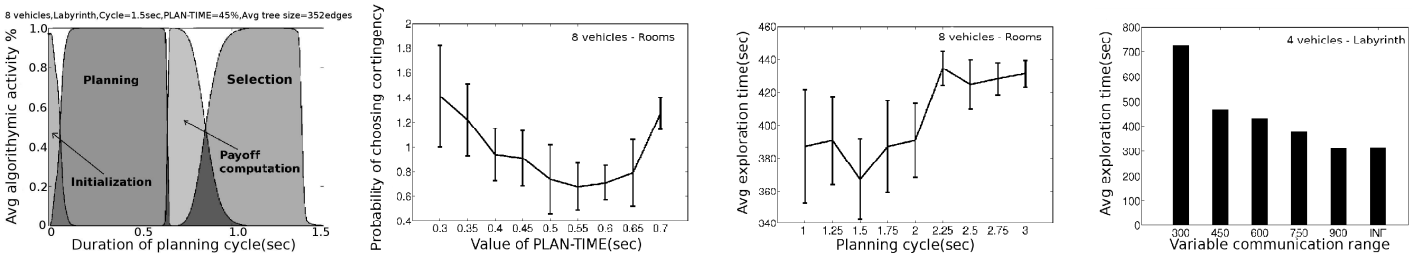

Fig. 7. Average activity profile during a cycle (left) and dependence on (from second to forth): `CYCLE_DURATION`, `PLAN_TIME`, and maximum communication range.

the second figure where we see the probability of executing contingency plans as a function of the portion of the planning cycle allocated to the motion planner (`PLAN-TIME`). This probability is very low ($0.7 - 1.5\%$). Moreover, there is an optimum value at around 55%. For small `PLAN-TIME`, the planner has little time to produce enough plans, while for larger ones max-plus has not enough time to make a selection and the contingency is selected for safety reasons. The third figure shows that increasing the duration of the planning cycle can result in performance deterioration. The last figure shows that as the communication range increases, four vehicles finish the exploration faster, which is expected.

V. Conclusions

This paper presents a novel integration of sampling-based planners with message-passing protocols for the distributed solution of planning problems that involve vehicles with dynamic constraints. It extends work on safe, real-time sampling-based planning to the case of multiple communicating vehicles. The method provides safety guarantees in terms of collision avoidance as well as in terms of retaining a connected communication network. The algorithm has been implemented on a distributed simulator and the results on vehicles with acceleration constraints confirm the safety properties of the approach in a worskspace exploration application. A comparison over priority-based schemes shows that the distributed protocol offers improved scalability.

The proposed method allows for plugging in other types of dynamic constraints and can also be integrated with higher-level approaches for distributed task assignment and distributed state estimation. Two important directions for improving the current framework is to study the effects of sensing uncertainty and limited communication reliability. We hope to address these issues and possible extensions in future work.

VI. Acknowledgments

Work on this paper 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.

References

[1] R. M. Murray, "Recent reseach in cooperative control of multi-vehicle systems," *(submitted) ASME Journal of Dynamic Systems, Measurement, and Control*, 2007.

[2] P. Ogren, E. Fiorelli, and N. E. Leonard, "Cooperative control of mobile sensor networks: Adaptive gradient climbing in distributed environments," *IEEE Tr. on Aut. Control*, vol. 49, no. 8, pp. 1292–1302, 2004.

[3] R. Olfati-Saber, "Flocking for multi-agent dynamic systems: Algorithms and theory," *IEEE Tr. on Aut. Control*, vol. 51, no. 3, pp. 401–420, 2006.

[4] G. A. S. Pereira, A. K. Das, V. Kumar, and M. F. M. Campos, "Decentralized motion planning for multiple robots subject to sesnsing and communication constraints," in *Work. on Multi-Robot Systems*, 2003.

[5] H. G. Tanner, G. J. Pappas, and V. Kumar, "Leader-to-formation stability," *IEEE TRA*, vol. 20, no. 3, June 2004.

[6] M. Egerstedt, X. Hu, and A. Stotsky, "Control of mobile platforms using a virtual vehicle approach," *IEEE Transactions on Automated Control*, vol. 46, no. 4, pp. 1777–1782, November 2001.

[7] L. Pallotino, V. G. Scordio, E. Frazzoli, and A. Bicchi, "Decentralized and scalable conflict resolution strategy for multi-agent systems," in *Int. Symp. on Mathematical Theory of Networks and Systems*, 2006.

[8] D. V. Dimarogonas, K. J. Kyriakopoulos, and D. Theodorakatos, "Totally distributed motion control of sphere world multi-agent systems using decentralized navigation functions," *ICRA*, 2006.

[9] S. Loizou, D. Dimarogonas, and K. Kyriakopoulos, "Decentralized feedback stabilization of multiple nonholonomic agents," in *ICRA*, vol. 3, 2004, pp. 3012–3017.

[10] S. LaValle, *Planning Algorithms*. Cambrdidge university Press, 2006.

[11] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE TRA*, vol. 12, no. 4, pp. 566–580, Aug. 1996.

[12] B. R. Donald, P. G. Xavier, J. F. Canny, and J. H. Reif, "Kinodynamic motion planning," *Journal of ACM*, vol. 40, no. 5, pp. 1048–1066, 1993.

[13] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *IJRR*, vol. 20, no. 5, pp. 378–400, May 2001.

[14] A. M. Ladd and L. E. Kavraki, "Motion planning in the presence of drift, underactuation and discrete system changes," in *RSS I*, Cambridge, MA, June 2005.

[15] K. E. Bekris and L. E. Kavraki, "Greedy but safe replanning under kinodynamic constraints," in *ICRA*, Rome, Italy, April 2007.

[16] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *ICRA*, 2006.

[17] M. Zucker, J. J. Kuffner, and M. Branicky, "Multipartite rrts for rapid replanning in dynamic environments," in *IEEE ICRA*, 2007.

[18] T. Fraichard and H. Asama, "Inevitable collision states - a step towards safer robots?" *Advanced Robotics*, vol. 18, no. 10, pp. 1001–1024, 2004.

[19] 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.

[20] P. Isto and M. Saha, "A slicing connection strategy for constructing prms in high-dimensional spaces," in *ICRA*, 2006.

[21] K. E. Bekris, K. I. Tsianos, and L. E. Kavraki, "A decentralized planner that guarantees the safety of communicating vehicles with complex dynamics that replan online," in *IROS (submitted)*, 2007.

[22] J. Pearl, *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[23] J. R. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," *Journal of Machine Learning Research*, vol. 7, pp. 1789–1828, 2006.

[24] K. Plarre and P. R. Kumar, "Extended message passing algorithm for inference in loopy gaussian graphical models," *Ad Hoc Networks*, vol. 2, pp. 153–169, 2004.

[25] C. Guestrin, D. Koller, and R. Parr, "Multiagent planning with factored mdps," in *NIPS-14*. MIT Press, 2002.

[26] G. Singh and A. J. Bernstein, "A highly asynchronous minimum spanning tree protocol," *Distributed Computing*, vol. 8, no. 3, pp. 151–161, March 1995.