

Real-Time Leader Election for Cooperative Control of Dynamic Vehicle Formations

Cristian Țăpuș
Computer Science Department
California Institute of Technology
crt@cs.caltech.edu

Lars B. Cremean
Control and Dynamics Systems
California Institute of Technology
lars@cds.caltech.edu

Abstract

In this paper we demonstrate the integration of real-time cooperative control of a set of dynamic vehicles with a leader election algorithm for wireless ad hoc networks. The algorithm makes use of spatial information and other application specific metrics to elect the best suited vehicle as the leader. Experimental results are provided that demonstrate integration of the algorithm in a real-time setting using gain-scheduled LQR control of hovercraft-like dynamical vehicles. We demonstrate real-time reconfiguration of the vehicle formation in the face of communication link creation and removal.

1 Introduction

Designing cooperative real-time control systems requires both a very thorough understanding of the dynamics of such systems and ways to coordinate individual components of the system to achieve the common goal through cooperation with other components. To dynamically choose a coordinator in such systems reduces to the problem of leader election.

The work presented in this paper focuses on the integration of a leader election algorithm for wireless ad hoc networks which makes use of spatial information and other application specific metrics with a cooperative control system. The algorithm guarantees that each set of agents which can communicate with each other will eventually elect a unique leader amongst themselves. The communication mechanism is restricted to directly connected neighbors and does not need global knowledge, making it suitable for mobile networks.

The algorithm is presented in the context of an experimental real-time control system called the Multi-Vehicle Wireless Testbed (MVWT) [3]. The testbed

consists of a number of vehicles that are each propelled over a smooth floor by high-powered ducted fans that are fixed to the vehicle. The vehicles each carry a small laptop for onboard computation, and are able to communicate with one another and with offboard command computers via 802.11b wireless Ethernet.

The introduction of multiple of these vehicles and communication between them enables broad and rich avenues of research in the areas of coordinated and cooperative control, and the testbed itself is a means by which new ideas in these areas can be validated and investigated in an environment which approximates many of the features of real world cooperative control applications. Applications relevant to this work include formation flight of unmanned aerial vehicles (UAVs), cooperative search and rescue operations, battlefield automation, swarm intelligence and autonomous exploration.

Given a cooperative control task, possibly motivated by the examples above, the premise is that we wish to organize the set of agents into a number of independent hierarchical groups, for which each individual has a clear notion of its “parent/child” relationship with other individuals in the group. The motivation for such a premise may come from communication or logistic constraints, for which a natural class of communication structure is described by tree-like graphs. In particular, it is known that under certain assumptions, agents’ connection in this manner satisfy the property of formation input-to-state stability (formation ISS) [5].

Work in leader election and routing algorithms for static environments has been studied in depth in the past [2]. Applying such algorithms to mobile networks and more dynamic environments have brought new challenges [4]. Proper control design that addresses these challenges allows for cooperative control systems that are both reconfigurable and robust to link and node failures as well as environmental changes.

2 Contributions

In contrast with our approach, previous work focuses on the success of the leader election process even at the cost of imposing restrictions on the motion of the agents or by assuming that network connectivity is somehow maintained (i.e. through fixed base stations). While that approach is suitable for sensor networks, static or dynamic clusters and similar environments, in cooperative control systems it is almost impossible to impose motion restriction on the agents. In the case of unmanned aerial vehicles, imposing motion requirements needed by the consensus algorithm might interfere with some of the safety real-time constraints imposed by the dynamics of the system. In this work, consensus and leader election algorithms are treated as tools, as opposed to being the goal of the system.

Previous implementations of leader-follower algorithms have required the role of each vehicle in the hierarchy to be hard-coded and static along the execution of a cooperative control task (e.g. formation flight). There are several advantages of the work presented here as compared to this paradigm. In particular, in this work the hierarchy of the formation is determined automatically and in a decentralized fashion based on a simple set of rules that are common to each vehicle in the formation. Additionally, this framework allows for event-based reconfiguration of the communication topology, so that the formation is robust to individual link or vehicle failures, or other external influences. Finally, allowing dynamic changes to the formation topology automatically enables the possibility for split and rejoin maneuvers and other emergent coordinated behavior.

3 Leader election algorithm

The system consists of n autonomous vehicles that communicate with each other through a wireless ad-hoc network. We represent the network as a graph with n nodes, where edges represent bi-directional communication channels. Furthermore, we consider a directed acyclic graph (DAG) obtained by orienting the edges of each connected component of the network graph such that each component contains a single node with no incoming edges, called the leader of the component. Each node in the graph keeps track of the incoming (parents) and outgoing (children) edges along with the identity of the *leader* of its component.

Our system is *event* driven. This means the system is considered stable until the occurrence of an event takes it into an unstable state. It then successfully recovers

and enters a new stable state.

The main assumption is that the system is only faced with one event at a time. We also assume that nodes have unique identifiers, communication channels are reliable and ordered and nodes can reliably detect failures of communication channels.

The invariants we want to maintain during our algorithm are the uniqueness of the leader in each component and the existence of one leader in each component.

3.1 Events

The events which drive our system are: direction change, node failure, connection failure, new connection, and new node.

The direction change event is triggered by the control side of the system. The node membership and topology of the graph are unchanged, but the spatial topology or the leader election metric changes, requiring the election of a new leader. This event also triggers the formation of the directed acyclic graph Γ_δ . The first step is the election of a new leader. The process is initiated at the current leader and is propagated in Γ_δ to all the nodes in the connected component during a “fan-out” phase. Candidates for the leader position are collected in a complementary manner from the leafs of Γ_δ and propagated up towards the leader (the “fan-in” phase). Proposals are refined at each node from the suggestions of all the children and finally, the current leader determines the identity of the new leader. The new DAG formation stage is done in a very similar manner and is initiated by the new leader. This stage can become more complex if the control system imposes certain constraints on the orientation of the network edges.

Node failure is very common in such mobile distributed environments. If such an event occurs, the DAG might have to be rebuilt. As per our assumptions, the neighbors of the dead node observe the event and take action as follows. All parents of the failed node delete it from their children list. The children that have at least one other parent, delete the node from their parents’ list. The orphans have to determine if they are still connected to the leader through their children or not. If there is no path from the orphan to the leader, in the undirected graph Γ associated with the network topology then the node failure partitioned the Γ . The orphan will have to initiate a new leader election algorithm in its own component. If the child and its leader are part of the same component the DAG has to be rebuilt. The “new DAG” stage is initiated by the current leader.

Due to the dynamics of mobile networks, communication channels can fail often. In this case we distinguish

two cases. One of the nodes of the failed communication channel becomes orphan. We fall back to the case presented above. Otherwise, nothing needs to be done since the connectivity of Γ_δ was preserved.

The system can also experience formation of new communication channels between two nodes. We distinguish again two cases. If both nodes have the same leader, the new edge added to Γ_δ has to be oriented in a way so it doesn't create any cycles. If the nodes have different leaders it means that two components are joining. In this case, the new leader is determined out of the two existing leader at the joining points and a new DAG is generated.

3.2 Quantitative analysis

We represent the number of nodes in the system by N and the number of connections between these nodes by E (the number of edges in the DAG). In terms of space, each node needs to keep track of its parents, its children and the leader. This makes the space required at each node, $O(N)$ times the size of the node information structure, which contains, besides the node id, spatial information specific to the application.

The number of messages sent during the execution of the algorithm has to be analyzed in the context of each possible event. The worst case scenario behavior for processing each event provides an upper bound on the communication complexity.

Most of the events can be processed through a fan-out followed by a fan-in phase, in the worst case. The complexity of the fan-in and the fan-out phases is $O(E)$. The node failure event can have a higher complexity if the leader is the one failing, since it requires the fan-out, fan-in procedure a number of times proportional to the number of orphans created by the leader's death. The upper bound on the communication complexity for this event is $O(E*N)$.

We can conclude that in the worst case a loose upper bound for the communication complexity of the algorithm is $O(E*N)$. We are currently investigating data representations and optimizations to reduce the communication complexity significantly, but that topic is beyond the scope of this paper.

4 System Description

The testbed used to implement these algorithms is the Caltech Multi-Vehicle Wireless Testbed (MVWT) [3], a platform for rapid-prototyping, implementation, development and verification of single- and multiple-vehicle

control algorithms. The MVWT vehicle is powered by two ducted fans that are fixed to the vehicle, and glides on low-friction omnidirectional ball casters. The resultant dynamics are underactuated, input-constrained, and second-order.

Each vehicle in the MVWT is capable of acting as a completely autonomous agent, receiving its position and orientation from an off-field vision processing computer that is connected to a set of four overhead cameras. Other inputs to each vehicle include gyroscopic sensors (to improve vehicle performance), optional onboard sonar sensors (for environment characterization), and any variety of communicated information from other vehicles on the field or from off-field command and control computers.

The control software for the MVWT is written on top of the open source RHexLib library and API [1]. RHexLib programs consist of a collection of C++ classes called `Modules` whose `update` methods execute at a fixed rate in an active state, and a `ModuleManager` that calculates a static schedule that ensures the continuous operation of each of the modules.

The resultant software architecture is highly modular, with built-in handling of interfaces to actuators, the vision system and onboard sensors, as well as utilities for user-defined inter-vehicle communication. It serves to make the MVWT a powerful and versatile tool for implementation of cooperative control strategies.

5 Experimental Results

We created a LQR controller for each vehicle in which we integrated the notions of parents and children. The parent sends the children some reference data to bring them in a wingman position. The child accepts that information and executes optimal control to the desired position along a circular trajectory.

The particular scenario considered here is the coordinated motion of a set of three vehicles for which there is rapid change in the communication topology. We demonstrate echelon formation attachment and split maneuvers as examples of the versatility of leader election strategies as applied to formation flight.

The vehicles start by following independent circular paths. They join and disperse as per the following events: a communication channel between vehicle 0 and vehicle 1 is created; vehicle 1 becomes the wingman of vehicle 0; a communication channel between vehicle 1 and vehicle 2 is created; vehicle 2 becomes the wingman of vehicle 1; communication between vehicle 0 and ve-

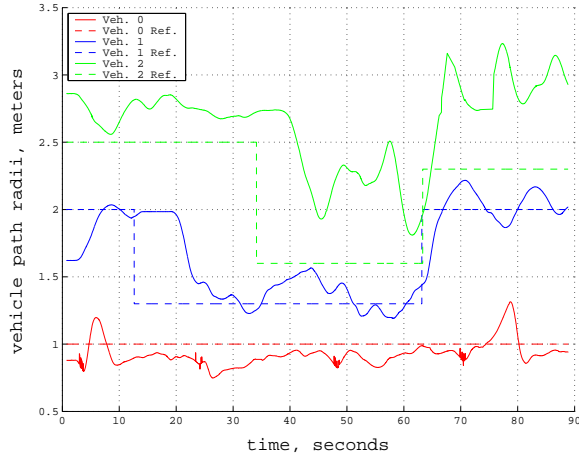


Figure 1. Distance of vehicles from center of reference trajectories. A link is created between vehicles 0 and 1 at 12 seconds, and between 1 and 2 at 34 seconds. The link between 0 and 1 is destroyed at 63 seconds. Vehicles are ordered from smallest to largest radius.

hicle 1 fails and vehicles 1 and 2 separate from vehicle 0 but vehicle 2 still remains the wingman of vehicle 1. During the time the three nodes were connected, vehicle 2 received commands from vehicle 0 via vehicle 1.

6 Summary and Future Directions

We have presented and implemented a leader election algorithm for use in general cooperative control problems involving independent autonomous dynamical agents. The algorithm utilizes a user-definable and distributable metric for determining leader follower relationships, which can be used for effective implementation of reconfigurable coordinated control maneuvers.

The purpose of the presented experimental results is to observe the behavior of the integrated leader election with the control system in a real life implementation. The response of the leader election algorithm to the events was prompt and did not interfere with the constraints of the control system in any way. The integration was seamless and it provided us with a testbed suited for far more complex applications than those we have done in the past.

Future work includes automating the creation of and elimination of communication links between agents depending on the current communication structure and on the spatial relationship between vehicles, refining our control algorithms for tighter performance, and developing a theory associated with combining the provable

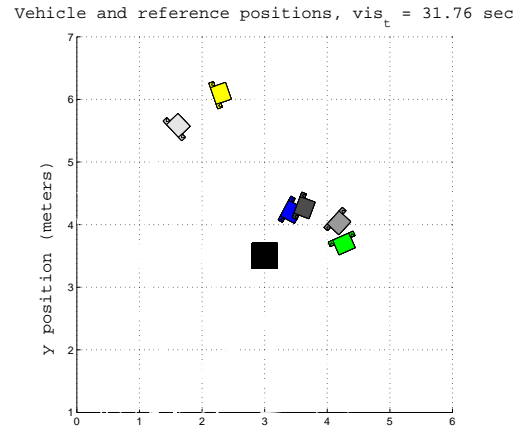


Figure 2. Top down snapshot of experimental data, taken at 31 seconds, after vehicle 0 (darkest, shown with reference) and 1 (farthest right) join but before they are joined by vehicle 2 (upper left). The vehicles each trail their reference slightly.

concepts of leader election from computer science with provable stability and performance results from control theory.

References

- [1] Rhexlib. URL: <http://sourceforge.net/projects/rhex/>.
- [2] *Distributed Computing: Fundamentals, Simulations and Advanced Topics*. McGraw-Hill, 1998.
- [3] Lars Cremean, William B. Dunbar, David van Gogh, Jason Hickey, Eric Klavins, Jason Meltzer, and Richard M. Murray. The Caltech multi-vehicle wireless testbed. In *Proc. of the 41st Conference on Decision and Control*, pages 86–88, 2002.
- [4] N. Malpani, J. Welch, and N. Vaidya. Leader election algorithms for mobile ad hoc networks, 2000.
- [5] Herbert G. Tanner, George J. Pappas, and Vijay Kumar. Input-to-state stability on formation graphs. In *Proc. of the 41st Conference on Decision and Control*, pages 2439–2444, 2002.