2003 American Control Conference
http://www.cds.caltech.edu/~murray/papers/2002t/dm03b-acc.html

The RoboFlag Competition

Raffaello D'Andrea
College of Engineering
Cornell University
rd28@cornell.edu

Richard M. Murray
Control & Dynamical Systems
California Institute of Technology
murray@cds.caltech.edu

Abstract

This paper provides an introduction to the RoboFlag competition and its use as a testbed for distributed control of multi-vehicle systems. The game consists of two teams of 6–10 robots and 2 people, each attempting to capture the other teams flag while defending their own. Limited sensing capability, distributed processing, and limited bandwidth communications are integral parts of the system. The features of RoboFlag provide a number of research challenges in distributed control, sensor fusion, team-based control, and human-centered control.

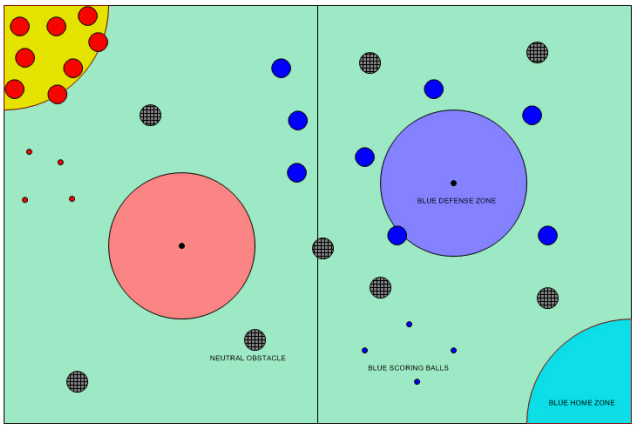

Figure 1: The RoboFlag playing field.

1 Introduction

RoboFlag is a game loosely based on "Capture the Flag" and "Paintball". Two teams play the game, the Red Team and the Blue Team. The Red Team's objective is to infiltrate Blue's territory, grab the Blue Flag, and bring it back to the Red Home Zone; concurrently, the Blue team's objective is to infiltrate Red's territory, grab the Red Flag, and bring it back to the Blue Home Zone. The game is thus a mix of offense and defense: secure the opponent's flag, while at the same time prevent the opponent from securing your flag. The field is depicted in Figure 1.

Points may be scored in several ways. The largest payoff occurs when an opponent's flag is safely brought back to the Home Zone. Points may also be scored by "tagging" an opponent in designated areas of the playing field. Points are lost when contact with "scoring balls" is made in an opponent's territory, and when contact with a neutral obstacle occurs. The game time is 40 minutes, with two 20 minute halves. There are no stops in play during each of the halves. Score keeping and time keeping are implemented via an autonomous Arbiter (the referee).

The game is meant to provide an abstraction of the systems captured in Figure 2. The system consists of a large number of vehicles in an environment. The vehicles must achieve a common objective: search and rescue, habitat monitoring, or a military scenario. Human operators are also present in the system. There is no specific relationship between the number of human operators and the number of vehicles. A centralized control unit may also be present, and serves to coordinate the vehicles. All these units can communicate with each other through a communications network, which is subject to bandwidth and latency limitations, service outages, etc. Finally, the global sensing unit provides global information to the entities (satellite information, GPS, etc.).

The research challenges associated with these types of systems is formidable. They range from the individual vehicle control, to the global planning and coordination of multi-agent systems. These challenges must be achieved in an uncertain, and often adversarial, environment.

This paper is an outline of the main features of this competition. The rules and regulations are constantly being revised and updated to reflect new research challenges, and to improve the game based on extensive game play. In the interest of clarity, we have decided to be as concrete as possible when describing the rules, even though these may have changed. The reader is referred to `http://roboflag.mae.cornell.edu` for an in-depth description of the current rules and regulations.

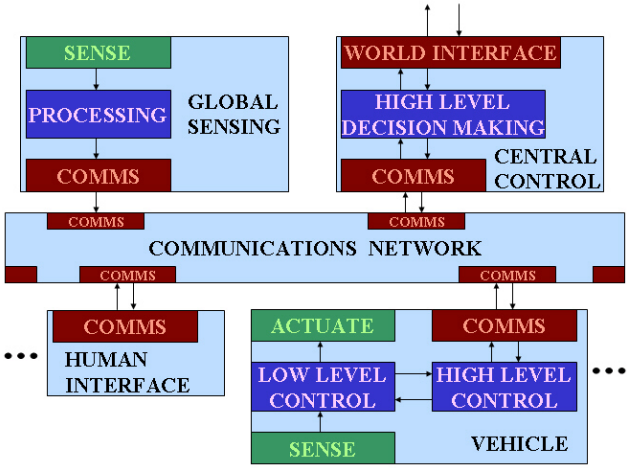

Figure 2: Systems of interest in RoboFlag.

The organization of the document is as follows. Section 2 gives a description of the playing field, the objects on the playing field, and the dynamics of the those objects. An overview of the information and control architecture for RoboFlag, and some of the research challenges associated with this, are given in Section 3. Section 4 gives a summary of the rules for the game. Finally, in Section 5 we give a short summary of some of the work that has been done on the testbed to date, including references to recent papers that give more detailed descriptions of various aspects of the RoboFlag competition.

2 RoboFlag Testbed

In this section we describe the RoboFlag testbed, focusing on the physical system. In the interest of clarity, we will be as concrete as possible in the description of the system. For example, we will assume that each team consists of eight robots and one central entity; clearly this can be changed. The parameter values referred to in this section are given in Table 1, at the end of the paper.

2.1 The Playing Field

The playing field, shown in Figure 1, consists of a FieldWidth by FieldLength carpeted area. The field is divided into two halves, each measuring FieldWidth by FieldLength/2, denoted the Blue Half and the Red Half. There are three zones in each half: the Home Zone, the Defense Zone, and the Attack Zone.

There is a coordinate system associated with the playing field, (x, y). The center of the playing field is at coordinates (0,0). The x coordinate is along the length of the playing field, and varies between -FieldLength/2 and FieldLength/2; the y coordinate is along the width of the playing field, and varies between -FieldWidth/2 and FieldWidth/2. The orientations of the Robots are in the range of $-\pi$ to π radians; counter-clockwise rotations from the x axis are positive, clockwise rotations are negative.

The coordinate system is not absolute, but relative to each team. For example, the coordinates of the center of the Blue Defense Zone in the Blue Team's coordinate system is the same as the coordinates of the center of the Red Defense Zone in the Red Team's coordinate system. For the remainder of this discussion, we will refer to the Blue Half in the diagram and express coordinates in the Blue Team's coordinate system; analogous definitions hold for the Red Half and the Red Team's coordinate system.

Home Zone. This area consists of a quarter circle of radius 1.0 meters. The coordinates of the center of the circle are (FieldLength/2, -FieldWidth/2), the corner of the field. Roughly speaking, the Blue Home Zone is a safe haven for the Blue Robots.

Defense Zone. The area consists of a circle of radius DefenseRad. The coordinates of the center of the circle are (DefenseX, DefenseY). Roughly speaking, the Blue Defense Zone is what the Blue Robots are trying to defend.

Attack Zone. This zone is the remainder of the half. Roughly speaking, the Blue Attack Zone is where the Blue Robots will attempt to stop the Red Robots from entering the Blue Defense Zone.

2.2 Objects on the Playing Field

During a game, the following objects can be on the playing field: 8 Red Robots, 8 Blue Robots, 8 Scoring Balls, and 8 Obstacles. Restrictions on the dynamics of these objects are outlined in Section 2.3. In the remainder of this document, all distances and locations are based on the centers of the objects.

Robots. The Robots conform to the RoboCup rules.[1] In particular, they fit inside a 0.18 meter diameter cylinder. The Robots are placed in their respective Home Zones at the beginning of the game.

Scoring Balls. The Scoring Balls are used to score points by shooting them at an opponent that is not in its own half. The Scoring Balls are golf balls. The balls are randomly placed in the Attack Zones (4 in each zone) at the beginning of the game.

Obstacles. Before the start of the game, 8 Obstacles are randomly placed on the playing field. The Obstacles are 0.20 meters in diameter. The restrictions on the initial Obstacle placement are as follows:

[1] http://www.robocup.org

1. The center of an Obstacle cannot be inside a Home Zone.
2. The separation between the centers of any two Obstacles must be at least D_{obSep}.

A uniform distribution is used to pick the location of the Obstacles on the playing field: if the chosen location is not allowed, a new location is chosen at random until all of the Obstacles are placed.

2.3 Dynamics

The dynamics of the game govern how the various objects on the playing field interact with each other and with the user. These dynamics have been chosen such that they are representative of the physical testbed environment.

Robots. The Robots are controlled by sending desired velocity commands directly to the Robots. Local control algorithms attempt to regulate the Robot velocities to the commanded Robot velocities. The local control algorithms have been developed so that the Robots satisfy the following simplified dynamic equations of motion:

$$\ddot{x}(t) = \alpha U_x(t) - \beta \dot{x}(t) \quad (1)$$

$$\ddot{y}(t) = \alpha U_y(t) - \beta \dot{y}(t) \quad (2)$$

$$\ddot{\theta}(t) = \alpha_\theta U_\theta(t) - \beta_\theta \dot{\theta}(t) \quad (3)$$

where $U_x^2 + U_y^2 \leq 1$ and $\text{abs}(U_\theta) \leq 1$.

It should be stressed that U_x and U_y **are not** the control inputs that are used to control the Robots; the Robots are controlled by directly sending velocity commands $\overline{V}_x$, $\overline{V}_y$, and $\overline{V}_\theta$ to the Robots. However, the velocity commands must be compatible with the dynamics described above in order for the Robot to actually track the velocity commands. For example, the Robot will not be able to track velocity commands that change drastically, as this would result in large values for the local control inputs U.

Velocity commands cannot be tracked instantaneously by the Robots. There is a delay of T_{delayOut} before the Robots can respond to velocity commands; this is further described in Section 3.1.

Scoring Balls. Collision of the Scoring Balls with the other objects on the field can be approximated as perfectly elastic. The mass of the Scoring Balls is much less than the mass of the Robots. The equation of motion of the ball when rolling can be approximated by

$$\ddot{z} = -\gamma \text{sign}(\dot{z}), \quad (4)$$

where z is along the direction of motion.

The Scoring Balls can be trapped and shot by the Robots. When a Robot issues a trap command, any Scoring Ball that makes contact with the Robot in a $\pi/2$ arc centered about the orientation of the Robot is kept attached to the Robot on the point of contact. A Robot must continually issue a trap command to ensure that a Scoring Ball remains attached to the Robot.

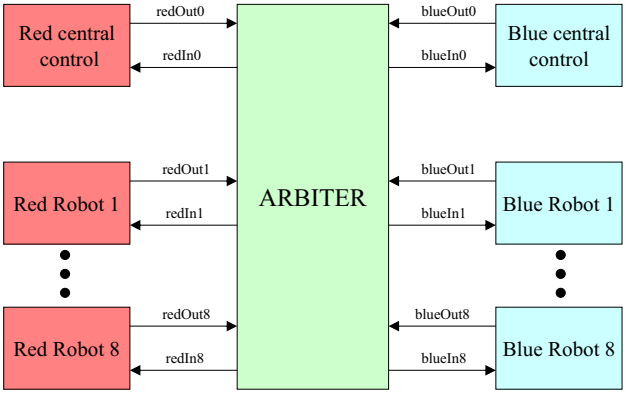

Figure 3: Information and Control Architecture

When a Robot issues a shoot command, any Scoring ball that makes contact with the Robot in a $\pi/2$ arc centered about the orientation of the Robot is shot away from the Robot *in the direction which coincides with the orientation of the Robot* at velocity V_{shoot} relative to the Robot. The shoot command overrides the trap command.

Obstacles. The Obstacles move throughout the game. A new random location for each Obstacle is determined at time t_{switch} from the previous switch time, where t_{switch} is a uniformly distributed random variable in the interval $[T_{\text{switchMin}}, T_{\text{switchMax}}]$. The first switch time occurs at time $t_{\text{switch}} = 0$. The Obstacles nominally move toward their new target position with maximum velocity V_{obs} If the Obstacles reach their destination before the next switch time, they remain motionless.

When an Obstacle-`inactive` Robot collision occurs, as determined by parameter D_{obs}, the target destination of the Obstacle is immediately set to its current destination.

3 Information and Control Architecture

The Information and Control Architecture is depicted in Figure 3. Each team is composed of 9 *Entities*: 8 Robot Entities and 1 Central Control Entity. All communication and control information for each team passes through the Arbiter.

Each Robot Entity receives local information from the Arbiter: its own position, orientation, translational and angular velocity, and game state, and the position, orientation, and game state of nearby objects. Each

Robot Entity sends local information through the arbiter: the desired Robot velocities, the shoot command (to shoot any Scoring Ball attached to the Robot), and the trap command (to trap any Scoring Ball that makes contact with the Robot in a 90 degree arc centered about the orientation of the Robot). As discussed in Section 2.3, all command information to the actual Robots is subject to a latency of T_{delayOut}. All incoming local information is also subject to latency, of value T_{delayIn}. Local information is passed to and from the Arbiter at a rate of FrameRate.

The Central Control Entity neither sends nor receives local information.

Each Entity can send and receive information to and from any other Entity via a communications network. Bandwidth and latency limitations on the communications network are described in Section 3.2.

A detailed description of the implementation of the Information and Control Architecture may be found in the RoboFlag Operations Documentation.[2]

3.1 Local Information

Each Robot Entity receives the following local information at a rate of FrameRate, subject to a latency of T_{delayIn}:

Own local information: the position, orientation, translational and angular velocity, and state of the Robot.

Nearby local information: the position, orientation, and state of all objects that satisfy the following conditions:

- are within a sector of radius $D_{\text{visRadius}}$ and of angular spread D_{visAngle} from the Robot (centered about the orientation of the Robot);
- a direct line of sight exists between the centers of the objects and the center of the Robot (the radii of objects are used to calculate if an occlusion is taking place).

Each Robot Entity sends the desired Robot velocity, shoot commands, and trap commands at a rate of FrameRate, subject to a latency of T_{delayOut}.

3.2 Communications Network

Each Entity can communicate with any other Entity via a Communications Network. The Communications Network is captured by the following parameters:

$B_{i,j}$**:** the maximum bandwidth from Entity i to Entity j;

$L_{i,j}$**:** the latency from Entity i to Entity j.

A message of size M is thus received by Entity j when a time $t = M/B_{i,j} + L_{i,j}$ has elapsed since Entity i sent the message, rounded up to the nearest multiple of frame time. If the bandwidth limits between two entities are exceeded, the packet is dropped from the outgoing message queue, resulting in loss of information.

3.3 Implications and Challenges

The information architecture described here requires that in order for the robots to maintain a shared situational awareness, they must communicate between each other and the Central Control Entity. Due to the limited communications rate and the limited information sensing, this represents a significant challenge in the high level control logic. There are a number of issues that arise.

Shared Situational Awareness. Since all robots have only local views of the game state, they must communicate with each other in order to take into account entities that are not within their sensing radius. This can be done either by maintaining a database in the Central Control Entity or by having reach robot maintain its own world state. Communication bandwidth limits require careful management of messages to ensure timely dissemination of information.

Information Fusion. In the current simulation environment, no sensor noise is present in the system. However, since the sensed position of a robot may be required at different times by different robots, they must still fuse their information to ensure that individual entities sensed by multiple robots are not counted as multiple objects.

Sensor Coverage. Robots must balance accomplishing their mission (offense or defense) with maintaining sensor coverage across the field. Ideally, robots should be placed so that they do not have large overlapping regions of visibility, while at the same time not leaving large gaps in coverage (where an opposing team member might slip through).

4 Rules of the Game

The rules that govern the RoboFlag game have been chosen to provide a challenging competition that forces the distributed, limited information aspects of the environment to be addressed. Modifications of these rules have been used in some versions of the competition to simplify the game play.

4.1 Robot States and State Transitions

For this discussion, we will refer to the Blue Robots; similar definitions hold for the Red Robots. Each

[2] `http://roboflag.mae.cornell.edu`

Robot is in one of four game states: `active`, `flagged`, `tagged`, or `inactive`. These states, and the transitions between states, are described below. The state transitions are also captured in Figure 4.

`active`: This is the normal operating state for a Robot, and is the default state when the game begins. When `active`, a Robot can tag other opponents for points, and capture the opponent's flag. A Blue Robot is set to the `active` state if

1. it is in the Blue Home Zone.

`flagged`: The Robot is in this state when it has captured the opponent's flag. Only one Robot per team can be in this state. The difference between the `active` and `flagged` states is that a `flagged` Robot can be tagged in more areas of the field. A Blue Robot becomes `flagged` if

2. no other Blue Robots are `flagged`, it is `active`, and the distance between it and the center of the Red Defense Zone is less than D_{flag}.

`tagged`: A Blue Robot becomes `tagged` in five ways:

3. When it is `active`, in the Red Attack Zone or the Red Home Zone, and an `active` Red Robot comes within D_{tagRobot} of it.
4. when it is `flagged`, anywhere on the playing field with the exception of the Blue Home Zone and the Blue Defense Zone, and an `active` Red Robot comes within D_{tagRobot} of it.
5. when it is `flagged`, anywhere in the Red Half, and a `flagged` Red Robot comes within D_{tagRobot} of it.
6. when it is `active` or `flagged`, anywhere on the playing field with the exception of the Blue Home Zone, and a Red Scoring Ball comes within D_{tagBall} of it.
7. when it is `active` or `flagged`, and it comes within D_{tagRobot} of *any* Robot that has been `inactive` or `tagged` for longer than T_{grace}.

A Robot in the `tagged` state cannot be controlled, cannot be communicated to, and cannot send information to the other Robots. When `tagged`, the Arbiter assumes control of the Robot and guides it back to the Robot's Home Zone with maximum velocity V_{tag}.

`inactive`: The Robot is in this state when it has violated one of the rules of the game. A Blue Robot is immediately deemed `inactive` if it is `active` or `flagged`, and any of the following conditions are satisfied:

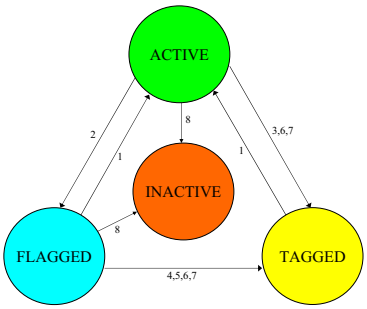

Figure 4: Transition between states

8a. it comes within D_{obs} of an Obstacle;

8b. if it enters the Blue Defense Zone;

8c. if it has traveled more than D_{refuel} since it was last in the Blue Home Zone.

An `inactive` Robot cannot be controlled, communicated to, and cannot send information to the other Robots for the remainder of the half. It remains stationary for the remainder of the half. The Arbiter will immediately take control of the Robot in question when `inactive`. When case **8b** occurs and the Robot is inside the Blue Defense Zone, the Arbiter will move the Robot to the closest point on the boundary of the Blue Defense Zone.

The above state transitions are captured in Figure 4.

4.2 Scoring Points

Points are assigned when transition between Robot states occur:

Tagged by an opponent: P_{opTag} points are assigned to the Blue Team when a Red Robot transitions to the `tagged` state via transitions **3,** **4**, or **5**, as described in Section 4.1.

Tagged by a ball: P_{ballTag} points are assigned to the Blue Team when a Red Robot transitions to the `tagged` state via transition **6**.

Tagged by an Arbiter controlled Robot: P_{arTag} points are assigned to the Blue Team when a Red Robot transitions to the `tagged` state via transition **7**.

Capturing the flag: P_{flagCap} points are assigned to the Blue Team when a Blue Robot transitions from the `active` state to the `flagged` state.

Bringing the flag home: P_{flagHome} points are assigned to the Blue Team when a Blue Robot transitions from the `flagged` state to the `active` state.

Points are also assigned at the end of the half based

on the number of inactive players on the playing field. P_{inactive} points are assigned to the Blue Team for every `inactive` Red Robot at the end of each half.

5 Summary and Work to Date

This paper provides a summary of the playing environment and rules for the RoboFlag competition. The competition is intended to stimulate research in areas of control related to the integration of communications, computing and control, as well as higher level decision making in feedback systems. It also explores issues related human-centered control and team-based motion planning algorithms.

An initial RoboFlag competition was held in Summer 2002 with two groups of students developing a user interface and programmable automation to compete in both simulated and experimental environments. A more complete description of the testbed hardware and software is given by D'Andrea and Babish [2]. A hierarchical, systems-based approach to the RoboFlag competition is described by Campbell et. al. [5]. The results from two teams who used the competition to develop human-in-the-loop, cooperative control strategies are described in the work of Chamberlain et al. [3] and Zigoris et al. [4]. An analysis of the Summer 2002 competition and a description of the future work is given by Adams and Hayes [1].

Acknowledgments: The following individuals have contributed to the development of the RoboFlag rules: Michael Babish, Mark Campbell, Matt Earl, Tama's Kalma'r Nagy, Andrey Klochko, Atif Chaudry, Jeff Fowler, Kent Cseh, Gonzalo Dominguez.

References

[1] J. A. Adams and A. T. Hayes. Surf competition: Results, analysis, and future work. In *Proc. American Control Conference*, 2003.

[2] M. Babish and R. D'Andrea. The RoboFlag testbed. In *Proc. American Control Conference*, 2003.

[3] L. Chamberlain, J. Tang, M. Watugala, J. A. Adams, and M. Babish. A behavioral architecture for strategy execution in the RoboFlag game. In *Proc. American Control Conference*, 2003.

[4] P. Zigoris, J. Siu, O. Wang, and A. T. Hayes. Balancing automated behavior and human control in multi-agent systems: a case study in RoboFlag. In *Proc. American Control Conference*, 2003.

[5] M. Campbell, R. D'Andrea, D. Schneider, A. Chaudhry, S. Waydo, J. Sullivan, J. Veverka, and A. Klochko. RoboFlag Games using Systems Based, Hierarchical Control. In *Proc. American Control Conference*, 2003.

Parameter	Description	Value
FieldWidth	width of field	4.0 m
FieldLength	length of field	6.0 m
DefenseRad	radius of Defense Zone	0.70 m
DefenseX	x-coord of Defense Zone	1.30 m
DefenseY	y-coord of Defense Zone	0.30 m
α	Robot translational acceleration coefficient	1.0 m/s^2
β	Robot translational velocity coefficient	1.0 1/s
α_θ	Robot rotational acceleration coefficient	5.0 rad/s^2
β_θ	Robot rotational velocity coefficient	5.0 1/s
γ	friction coefficient for ball	0.25 m/s^2
T_{delayOut}	latency of Robot commands	0.060 s
T_{delayIn}	latency of incoming local information	0.10 s
$T_{\text{switchMin}}$	minimum Obstacle switch time	10 s
$T_{\text{switchMax}}$	maximum Obstacle switch time	30 s
T_{grace}	grace period for `inactive` and `tagged` Robots	5 s
D_{obSep}	minimum Obstacle-Obstacle steady state separation	0.70 m
D_{obCol}	Obstacle-Obstacle collision separation	0.25 m
D_{obs}	Obstacle-Robot separation for `inactive` determination	0.24 m
D_{refuel}	maximum distance before refueling	20 m
D_{tagRobot}	distance for Robot-Robot tag	0.23 m
D_{tagBall}	distance for Robot-Scoring Ball tag	0.16 m
D_{flag}	distance for Robot-Flag capture	0.05 m
$D_{\text{visRadius}}$	radius of visibility sector	0.60 m
D_{visAngle}	spread of visibility sector	pi/2 rad
V_{shoot}	relative speed of shot Scoring Ball	1.00 m/s
V_{obs}	speed of Obstacles	0.05 m/s
V_{tag}	speed of tagged Robots	0.10 m/s
FrameRate	system frame rate	30 fps
$B_{i,j}$	communications bandwidth	1024 bps
$L_{i,j}$	communications latency	0.20 s
P_{opTag}	points for Robot-Robot tag	1
P_{ballTag}	points for a Scoring Ball-Robot tag	5
P_{arTag}	points for an Arbiter-Robot tag	10
P_{flagCap}	points for capturing Flag	5
P_{flagHome}	points for bringing Flag home	25
P_{inactive}	points for `inactive` Robots	10

Table 1: Parameter values for RoboFlag