University POLITEHNICA of Bucharest, Romania

Faculty of Automatic Control and Computer Science

# Proiectarea Algoritmilor 2011

Server Manual

**Document version**: 1.1

# 1. Overview

The purpose of this document is to present the Server module for the car race simulation project. It will be used as a manual for installing and deploying the Server, connecting with the Viewer, developing the AIs and finally, connecting with the AIs.

This document is intended for the development teams participating and will explain in detail the entire Server – AI protocol. Some of the features outlined below may not be of use to some teams and thus may be ignored. They are only mentioned for the sake of completion.

## 1.1 Server Characteristics

### Event Based Implementation

- request-response behavior

### Highly Scalable

- multithreaded implementation
- the Viewer and AIs can be connected in parallel, no constraints

### Network Configurable

- binds to hostname and port set by user

### Multiple Races allowed

- no need to restart
- restart policy on errors

### AI controller

- user controlled car for testing server

### Console Input

- user can ask the server for information
- ability to force race start if some AIs fail

### Message Queue drop policy

- the AI can specify if he wants to drop the message queue

### Map compression

- the map is compressed for speeding communication with AIs

### High-Precision distance

- line-cross algorithm for calculating distance traveled on track

### Flexible AI protocol

- messages have headers for maintaining compatibility with legacy versions

### Debug-Mode

- if wanted, all the messages exchanged can be viewed on console

### Wrong Way notification

- the AI is informed if he’s going the right way

### Log Files (next release)

- all the status and error messages are logged into hard-drive

### Tournament Results caching (next release)

- store the tournament info on the server

## 1.2 Step-by-step install & run

The server module the next files:

***/server.jar*** – server binaries

***/server.bat*** – server executable

***/aiController.jar*** – aiController binaries

***/high_speed_car.bat*** – aiController executable

***/default_car.bat*** – aiController executable

***/best_handling_car.bat*** – aiController executable

In order to configure the server, edit the ***server.bat*** file. The parameters are:

- *hostname* to bind to (egs. *localhost*)
- *port* to listen for viewer connections (egs. *4444*)
- *port* to listen for AI connections (egs. *4445*)
- *debug* to show communication on console (egs. *false*)

Double-click on ***server.bat*** to start the server.

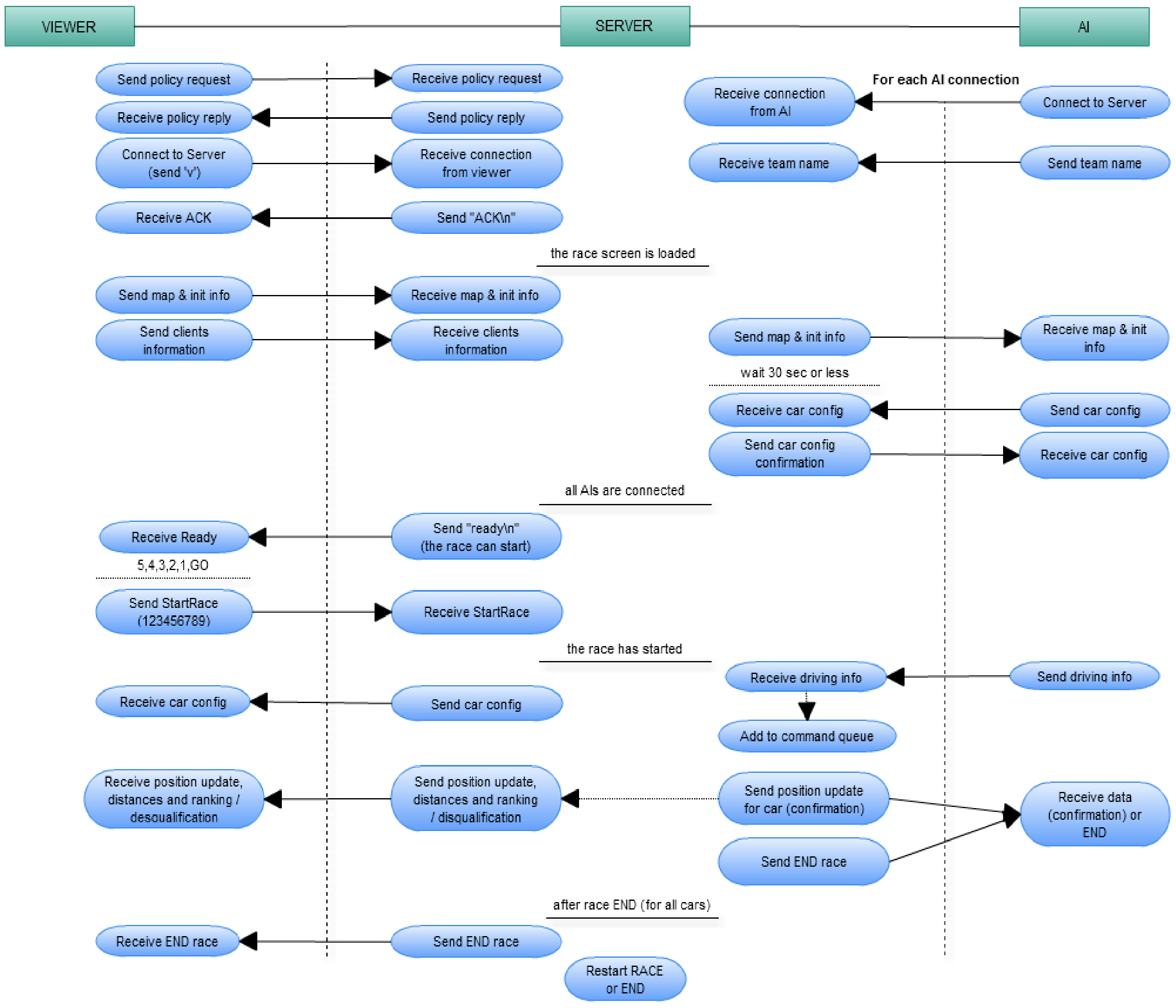
AI controller can be used for playing a car from keyboard, by using the arrows. Before starting the controller, you should configure it properly.

The parameters for AI controller are:

- *hostname* to connect to (egs. *localhost*)
- *port* to connect to server (egs. 4444)
- *team_name* for the current race, as stated by the Viewer (egs. WackyRacers)

# 2. Development Manual

## 2.1 AI – Protocol Diagram



## 2.2 Connect AI with Server – Step By Step

All data should be sent/received in big endian format. Strings should be sent/received byte with byte, in UTF-8 encoding. One integer has 4 bytes. One double has 8 bytes.

Each message from the AI-Server network protocol has a 1 integer header. This header represents the message type and the Server expects different data, based upon this value.

The messages should be sent / received exactly in the order displayed on the diagram.

### Headers

[1] Team name
[2] Map dimensions
[3] Initial info
[4] Car configuration (and confirmation)
[5] Send driving info
[6] Receive position confirmation
[7] End Race

### Send team name

- header (1 integer)
- name length (1 integer)
- case-sensitive name (N * 1 byte)

The AI needs to correctly send the team name in order to connect to a race, because the Server makes the associations AI – car based upon the team name received from Viewer.

### Receive map

- header (1 integer)
- width in pixels (1 integer)
- height in pixels (1 integer)
- map (width*height + padding integers)

The encoding for the map: 32 values inside one integer. If the map is not 32 multiple, then 0 padding is used.

## Receive initial info

- header (1 integer)
- start point X (1 integer)
- start point Y (1 integer)
- width in meters (1 integer)
- height in meters (1 integer)
- direction (1 integer)
- number of laps (1 integer)
- maximum lap time (1 integer)
- initial car angle (1 double)

The direction is 1 for clockwise, 0 for counter-clockwise. The maximum lap time is represented in miliseconds. The initial car angle is a value in radians in [0:2*PI].

## Send car configuration

- header (1 integer)
- max acceleration in m/s*s (1 integer)
- max brake in m/s*s (1 integer)
- max top speed in m/s*s (1 integer)
- steering speed in rad/s (1 double)

The initial steering speed is a value in radians in [0:2*PI]. These values should respect the formula **10*ACC+10*BRK+SPD+200*STR = 400.** This message has a confirmation. If the formula is not respected, the Server responds with the default values (ACC = 10, BRK = 10, SPD = 100, STR = 0.5) .

## Receive car configuration confirmation

- header (1 integer)
- max acceleration in m/s*s (1 integer)
- max brake in m/s*s (1 integer)
- max top speed in m/s*s (1 integer)
- steering speed in rad/s (1 double)

The header has the same value as the previous message, which is 4.

## Send driving info

- header (1 integer)
- acceleration percentage (1 integer)
- brake percentage (1 integer)
- wanted angle (1 double)
- drop queue (1 integer)

The value for drop queue should be 1, if and only if the AI wants to drop all the messages he has sent but haven’t been applied yet. Else, this value should be 0.

## Receive position confirmation

- header (1 integer)
- current X in meters (1 double)
- current Y in meters (1 double)
- current speed in m/s (1 double)
- current angle in radians (1 double)
- current direction (1 integer)

The current direction is 1, if the AI is driving correctly in the race. If he’s driving reverse, -1 is sent.

## Receive end race

- header (1 integer)