No Tipping Protocol

Hunter Freyer

The server which will be used to judge the No Tipping Game in the Heuristics Problem Solving class will communicate with each of the individual strategy clients over a TCP connection, on port 10987. Presumably, both client and server will be running on the same machine, so use your favorite language to connect to:

localhost:10987

The client and server will interact with a very simple protocol, defined in this document. The protocol uses simple and stupid syntax, reminiscent of /etc/passwd format. It goes something like this:

COMMAND:ARG1:ARG2:ARG3;

Client and server will send commands of this form one at a time, with the COMMAND being a single word. Spacing and line breaks are irrelevant - all that matters is the colons and semicolons. Let's take a look at how this conversation will play out.

Immediately after successfully connecting over TCP, the client sends this to the server:

JOINREQUEST:<handle>:<playernumber>;

The handle is the name you'll be known by in the game, and the player number is either 0 or 1. The client decides which player number it wants to have so your program should accept an argument for whether you want to go first or second. Here's an example of the above command:

JOINREQUEST:hjfreyer:0;

Would attempt to register hjfreyer as player 0 (i.e. the first player).

The server responds to this request with either:

GRANTED;

or

DENIED;

Denied is received if that slot has already been taken by another player. The client should then wait for the game to begin. When the game begins, the server sends the following to both players:

BEGIN:<gamenumber>:<maxweight>;

The game number is just something for easy reference, and isn't important. The maxweight is the maximum value of the weights. In the example of the problem shown in class, the maxweight was 7. Maxweight will never be greater than 10.

After this notification, the players both wait for the server to tell them it's their turn with the ROUND command. There are two versions of the round command, one when adding weights, and one when removing. Here's the version when adding:

ROUND:<round number>:ADD:
<your weights>:
<their weights>:
<game state>;

In the above command, the round number is a number of the form X.Y, where X is round number and Y is the number of the current player (0 or 1).

Your weights will be a set of numbers corresponding to which weights you haven't placed yet. Same with their weights. The order of these two lines is INDEPENDENT of whether you go first or second. Yours are always first, theirs are always second.

The game state will consist a list of 21 numbers, corresponding to marked positions on the bar, starting at -10, going up through 10. If the number is a zero, then there is no weight at that location. If the number is nonzero, then there is a weight at that location, with the given value. Let's see an example just to be clear:

ROUND:2.0:ADD:
1 2 3 6 7:
2 3 4 5 6 7:
0 0 0 0 5 3 0 0 1 0 0 0 4 0 0 0 0 0 0 0 0;

This is round 2, player 1's turn. We're still adding weights. You have weights 1, 2, 3, 6, 7 to place. You get the idea.

If we're removing, the command looks slightly different:

ROUND:8.1:REMOVE:
0 0 0 0 5 3 0 0 1 0 0 0 4 0 0 0 0 0 0 0 0;

It's exactly the same, but no weights are given, because both players have spent all of them.

The client then responds with, depending on whether we're adding or removing,

either:

ADD:3:-2;

Where 3 is the weight number, and -2 is the index at which to place it, or

REMOVE:-2;

Where -2 is the index of the weight to remove.

If this move is not valid for whatever reason (space already taken, weight already used, etc), then the game is considered forfeit, and the server sends back:

INVALID;

The game proceeds this way, each client waiting for a ROUND command, and replying with an ADD or REMOVE, until eventually the game ends, at which point

WIN;

is sent to the winner, and

LOSE;

to the... 2nd winner.