universidade de aveiro

Sistemas Operativos (P1)

Simulação de jogo de Futebol

Licenciatura em Engenharia Informática

Afonso Boto nº89285

Tomás Candeias nº89123

2020/2021

Introdução

Jogo de futebol que consiste em quatro jogadores e um guarda redes por equipa com a arbitragem de um árbitro, todos os outros estarão a mais e serão assinalados com o estado LATE.

O árbitro acaba por ser uma peça fundamental, pois é ele que inicia a partida e que a termina.

Será abordado ao longo deste relatório todos os acréscimos no código fonte, fornecido pelo professor, bem como os testes e a validação.

semSharedMemReferee.c

arrive()

```
static void arrive ()
{
    if (semDown (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }

    /* TODO: insert your code here */
    sh->fSt.st.refereeStat = ARRIVING;
    saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }

    usleep((100.0*random())/(RAND_MAX+1.0)+10.0);

}
```

Adicionámos na zona crítica uma mudança de estado do árbitro para ARRIVING e guardámos o estado.

waitForTeams()

```
static void waitForTeams ()
{
    if (semDown (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }

    // esperar pelo estado
    sh->fSt.st.refereeStat = WAITING_TEAMS;
    saveState(nFic, &sh->fSt);


    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }

    if (semDown(semgid, sh->refereeWaitTeams) == -1 ){
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }
}
```

Adicionámos na zona crítica a mudança de estado do árbitro para WAITING_TEAMS e gravámos o estado. Posteriormente, já fora da zona crítica, colocamos um semáforo down, esperando que os jogadores e guarda redes formem as equipas.

startGame()

```
static void startGame ()
{
    if (semDown (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }


    sh->fSt.st.refereeStat = STARTING_GAME;
    saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }

    /* TODO: insert your code here */
    for(int i = 0; i < NUMPLAYERS; i++){
        if (semUp (semgid, sh->playersWaitReferee) == -1) {
            perror ("error on the up operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
        }
    }

}
```

Adicionámos na zona crítica a mudança de estado do árbitro para STARTING_GAME e gravámos o estado. Posteriormente, já fora da zona crítica, o for loop com a utilização de um semáforo up tem como objetivo avisar todos os jogadores de que o jogo vai começar.

play()

```
static void play ()
{
    if (semDown (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }

    sh->fSt.st.refereeStat = REFEREEING;
    saveState(nFic, &sh->fSt);
    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }

    usleep((100.0*random())/(RAND_MAX+1.0)+900.0);
}
```

Adicionámos na zona crítica a mudança de estado do árbitro para REFEREEING e gravámos o estado.

endGame()

```
static void endGame ()
{
    if (semDown (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }

    sh->fSt.st.refereeStat = ENDING_GAME;
    saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the up operation for semaphore access (RF)");
        exit (EXIT_FAILURE);
    }

    for(int i = 0; i < NUMPLAYERS; i++){
        if (semUp (semgid, sh->playersWaitEnd) == -1) {
            perror ("error on the up operation for semaphore access (RF)");
            exit (EXIT_FAILURE);
        }
    }

}
```

Adicionámos na zona crítica a mudança de estado do árbitro para ENDING_GAME e gravámos o estado. Posteriormente, já fora da zona crítica, o for loop com a utilização de um semáforo up tem como objetivo avisar todos os jogadores de que o jogo vai acabar.

semSharedMemGoalie.c

arrive()

```
static void arrive(int id)
{
    if (semDown (semgid, sh->mutex) == -1)  {
        perror ("error on the up operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }

    sh->fSt.st.goalieStat[id] = ARRIVING;
    saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the down operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }

    usleep((200.0*random())/(RAND_MAX+1.0)+60.0);
}
```

Adicionámos na zona crítica a mudança de estado do guarda redes para ARRIVING e gravámos o estado.

goalieConstituteTeam()

```
static int goalieConstituteTeam (int id)
{
    int ret = 0;
    int check = 0;
    if (semDown (semgid, sh->mutex) == -1)  {
        perror ("error on the up operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }

    sh->fSt.goaliesArrived++;
    sh->fSt.goaliesFree++;
    if (sh->fSt.goaliesArrived > 2){
        sh->fSt.st.goalieStat[id] = LATE;
        saveState(nFic, &sh->fSt);

    }
    else {

        if ( sh->fSt.playersFree < 4){
            // waiting team
            check = 1;
            sh->fSt.st.goalieStat[id] = WAITING_TEAM;
            saveState(nFic, &sh->fSt);


        }
        else{
            // forming team

            sh->fSt.st.goalieStat[id] = FORMING_TEAM;
            sh->fSt.goaliesFree--;
            ret = sh->fSt.teamId;
            saveState(nFic, &sh->fSt);
            for(int i = 0; i < NUMTEAMPLAYERS; i++){
                if ( semUp(semgid, sh->playersWaitTeam) == -1){
                    perror ("error on the up operation for semaphore access (GL)");
                    exit (EXIT_FAILURE);
                }

                if ( semDown(semgid, sh->playerRegistered) == -1){
                    perror ("error on the up operation for semaphore access (GL)");
                    exit (EXIT_FAILURE);
                }

            }
```

```
            sh->fSt.playersFree = sh->fSt.playersFree - NUMTEAMPLAYERS;
            sh->fSt.teamId++;
            saveState(nFic, &sh->fSt);


            if ( semUp(semgid, sh->refereeWaitTeams) == -1){
                perror ("error on the up operation for semaphore access (GL)");
                exit (EXIT_FAILURE);
            }
        }
    }


    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the down operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }
    if( check){
        if ( semDown(semgid, sh->goaliesWaitTeam) == -1){
            perror ("error on the up operation for semaphore access (GL)");
            exit (EXIT_FAILURE);
        }

        ret = sh->fSt.teamId;
    }

    if ( semUp(semgid, sh->playerRegistered) == -1){
        perror ("error on the up operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }


    return ret;
}
```

Adicionámos, dentro da zona crítica, a incrementação dos guarda redes que chegaram e disponíveis nas variáveis correspondentes.

Fizemos uma estrutura de controlo para identificar, em primeiro caso, se o guarda redes passado como argumento na função chegou atrasado. Numa segunda condição, para um guarda redes que fica no estado WAITING_TEAM, pois neste caso não existem jogadores disponíveis para formar equipa. Por último, no terceiro caso, há capacidade de formar equipa e o guarda redes passa para o estado FORMING_TEAM e forma a equipa.

waitReferee()

```
static void waitReferee (int id, int team)
{
    if (semDown (semgid, sh->mutex) == -1)  {
        perror ("error on the up operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }
    if(team == 1)
        sh->fSt.st.goalieStat[id] = WAITING_START_1;
    else if(team == 2)
        sh->fSt.st.goalieStat[id] = WAITING_START_2;

    saveState(nFic, &sh->fSt);



    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the down operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }

    if (semDown (semgid, sh->playersWaitReferee) == -1) {
        perror ("error on the down operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }

}
```

Adicionámos na zona crítica dependo da equipa do guarda redes a mudança de estado para WAITING_START_X, sendo o X passado como argumento na função (int team), e gravámos o estado.

Posteriormente, já fora da zona crítica, temos um semáforo down para que o guarda redes espere pelo início do jogo por parte do árbitro.

playUntilEnd()

```
static void playUntilEnd (int id, int team)
{
    if (semDown (semgid, sh->mutex) == -1)  {
        perror ("error on the up operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }

    if(team == 1)
        sh->fSt.st.goalieStat[id] = PLAYING_1;
    else if(team == 2)
        sh->fSt.st.goalieStat[id] = PLAYING_2;

    saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the down operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }

    if (semDown (semgid, sh->playersWaitEnd) == -1) {
        perror ("error on the down operation for semaphore access (GL)");
        exit (EXIT_FAILURE);
    }

}
```

Adicionámos na zona crítica dependo da equipa do guarda redes a mudança de estado para PLAYING_X, sendo o X passado como argumento na função (int team), e gravámos o estado.

Posteriormente, já fora da zona crítica, temos um semáforo down para que o guarda redes espere pelo final do jogo por parte do árbitro.

semSharedMemPlayer.c

arrive()

```
static void arrive(int id)
{
    if (semDown (semgid, sh->mutex) == -1)  {
        perror ("error on the up operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }

    sh->fSt.st.playerStat[id] = ARRIVING;
    saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the down operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }

    usleep((200.0*random())/(RAND_MAX+1.0)+50.0);
}
```

Adicionámos na zona crítica a mudança de estado do jogador para ARRIVING e gravámos o estado.

playerConstituteTeam()

```
static int playerConstituteTeam (int id)
{
    int ret = 0;
    int check = 0;
    if (semDown (semgid, sh->mutex) == -1)  {
        perror ("error on the up operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }


    sh->fSt.playersArrived++;
    sh->fSt.playersFree++;
    if (sh->fSt.playersArrived > 8){
        sh->fSt.st.playerStat[id] = LATE;
        saveState(nFic, &sh->fSt);

    }
    else {

        if ( sh->fSt.playersFree < 4 || sh->fSt.goaliesFree < 1 ){
            // waiting team
            check = 1;
            sh->fSt.st.playerStat[id] = WAITING_TEAM;
            saveState(nFic, &sh->fSt);


        }
        else{
            // forming team

            sh->fSt.st.playerStat[id] = FORMING_TEAM;
            sh->fSt.playersFree--;
            ret = sh->fSt.teamId;
            saveState(nFic, &sh->fSt);
            for(int i = 0; i < NUMTEAMPLAYERS - 1; i++){
                if ( semUp(semgid, sh->playersWaitTeam) == -1){
                    perror ("error on the up operation for semaphore access (PL)");
                    exit (EXIT_FAILURE);
                }

                if ( semDown(semgid, sh->playerRegistered) == -1){
                    perror ("error on the up operation for semaphore access (PL)");
                    exit (EXIT_FAILURE);
                }


            }
```

```
            if ( semUp(semgid, sh->goaliesWaitTeam) == -1){
                perror ("error on the up operation for semaphore access (PL)");
                exit (EXIT_FAILURE);
            }

            if ( semDown(semgid, sh->playerRegistered) == -1){
                perror ("error on the up operation for semaphore access (PL)");
                exit (EXIT_FAILURE);
            }

            sh->fSt.playersFree = sh->fSt.playersFree - NUMTEAMPLAYERS + 1;
            sh->fSt.goaliesFree--;
            sh->fSt.teamId++;


            if ( semUp(semgid, sh->refereeWaitTeams) == -1){
                perror ("error on the up operation for semaphore access (PL)");
                exit (EXIT_FAILURE);
            }
        }
    }

    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the down operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }

    if( check){
        if ( semDown(semgid, sh->playersWaitTeam) == -1){
            perror ("error on the up operation for semaphore access (PL)");
            exit (EXIT_FAILURE);
        }

        ret = sh->fSt.teamId;
    }

    if ( semUp(semgid, sh->playerRegistered) == -1){
        perror ("error on the up operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }



    return ret;
}
```

Adicionámos, dentro da zona crítica, a incrementação dos jogadores que chegaram e disponíveis nas variáveis correspondentes.

Fizemos uma estrutura de controlo para identificar, em primeiro caso, se o jogador passado como argumento na função chegou atrasado. Numa segunda condição, para um jogador que fica no estado WAITING_TEAM, pois neste caso não existem jogadores/guarda redes disponíveis para formar equipa. Por último, no terceiro caso, há capacidade de formar equipa e o jogador passa para o estado FORMING_TEAM e forma a equipa.

waitReferee()

```
static void waitReferee (int id, int team)
{
    if (semDown (semgid, sh->mutex) == -1)  {
        perror ("error on the up operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }

    if(team == 1)
        sh->fSt.st.playerStat[id] = WAITING_START_1;
    else if(team == 2)
        sh->fSt.st.playerStat[id] = WAITING_START_2;

    saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the down operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }

    if (semDown (semgid, sh->playersWaitReferee) == -1) {
        perror ("error on the down operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }

}
```

Adicionámos na zona crítica dependo da equipa do jogador a mudança de estado para WAITING_START_X, sendo o X passado como argumento na função (int team), e gravámos o estado.

Posteriormente, já fora da zona crítica, temos um semáforo down para que o jogador espere pelo início do jogo por parte do árbitro.

playUntilEnd()

```
static void playUntilEnd (int id, int team)
{
    if (semDown (semgid, sh->mutex) == -1)  {
        perror ("error on the up operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }

    if(team == 1)
        sh->fSt.st.playerStat[id] = PLAYING_1;
    else if(team == 2)
        sh->fSt.st.playerStat[id] = PLAYING_2;

    saveState(nFic, &sh->fSt);

    if (semUp (semgid, sh->mutex) == -1) {
        perror ("error on the down operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }

    if (semDown (semgid, sh->playersWaitEnd) == -1) {
        perror ("error on the down operation for semaphore access (PL)");
        exit (EXIT_FAILURE);
    }

}
```

Adicionámos na zona crítica dependo da equipa do jogador a mudança de estado para PLAYING_X, sendo o X passado como argumento na função (int team), e gravámos o estado.

Posteriormente, já fora da zona crítica, temos um semáforo down para que o jogador espere pelo final do jogo por parte do árbitro.

Testes e Validação

Teste do Referee

P00	P01	P02	P03	P04	P05	P06	P07	P08	P09	G00	G01	G02	R01
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	1	1	1	1	0	0	0	0	0	2	0	0	0
1	1	1	1	1	0	0	0	0	0	2	0	0	0
1	1	1	1	1	0	0	1	0	0	2	0	0	0
1	1	1	1	1	0	0	1	0	0	2	0	0	0
1	1	1	1	1	0	0	1	0	0	2	1	0	0
1	1	1	1	1	0	0	1	1	0	2	1	0	0
1	1	1	1	1	2	0	1	1	0	2	1	0	0
1	1	1	1	1	2	0	1	1	7	2	1	0	0
1	1	3	1	1	2	0	1	1	7	2	1	0	0
1	3	3	1	1	2	0	1	1	7	2	1	0	0
1	3	3	3	1	2	0	1	1	7	2	1	0	0
1	3	3	3	3	2	0	1	1	7	2	1	0	0
1	3	3	3	3	2	0	1	1	7	3	1	0	0
1	3	3	3	3	2	0	1	1	7	3	1	7	0
1	3	3	3	3	2	0	1	1	7	3	1	7	1
1	3	3	3	3	2	7	1	1	7	3	1	7	1
1	3	3	3	3	2	7	1	1	7	3	4	7	1
4	3	3	3	3	2	7	1	1	7	3	4	7	1
4	3	3	3	3	2	7	4	1	7	3	4	7	1
4	3	3	3	3	2	7	4	4	7	3	4	7	1
4	3	3	3	3	4	7	4	4	7	3	4	7	1
4	3	3	3	3	4	7	4	4	7	3	4	7	2
4	3	5	3	3	4	7	4	4	7	3	4	7	2
4	5	5	3	3	4	7	4	4	7	3	4	7	2
4	5	5	5	3	4	7	4	4	7	3	4	7	2
4	5	5	5	5	4	7	4	4	7	3	4	7	2
4	5	5	5	5	4	7	4	4	7	5	4	7	2
6	5	5	5	5	4	7	4	4	7	5	4	7	2
6	5	5	5	5	4	7	6	4	7	5	4	7	2
6	5	5	5	5	4	7	6	4	7	5	4	7	3
6	5	5	5	5	4	7	6	6	7	5	4	7	3
6	5	5	5	5	6	7	6	6	7	5	4	7	3
6	5	5	5	5	6	7	6	6	7	5	6	7	3
6	5	5	5	5	6	7	6	6	7	5	6	7	4

Neste teste de validação do árbitro fomos verificar se passava para o estado STARTING_GAME (2) quando as equipas já estavam formadas.

Deste modo, fomos verificar também se no estado REFEREEING (3) as equipas já apresentavam o estado PLAYING_1/PLAYING_2 (5/6).

Teste do Goalie

P00	P01	P02	P03	P04	P05	P06	P07	P08	P09	G00	G01	G02	R01
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0	1	0	0	0
0	1	0	1	1	0	0	0	0	0	1	0	0	0
0	1	0	1	1	0	0	0	0	0	1	0	0	0
0	1	0	1	1	0	0	0	0	0	1	1	0	0
0	1	2	1	1	0	0	0	0	0	1	1	0	0
0	1	2	1	1	0	0	1	0	0	1	1	0	0
0	1	2	1	1	0	1	1	0	0	1	1	0	0
0	1	2	1	1	1	1	1	0	0	1	1	0	0
0	1	2	1	1	1	1	1	0	2	1	1	0	0
0	1	2	1	1	1	1	1	0	2	1	1	7	0
7	1	2	1	1	1	1	1	0	2	1	1	7	0
7	1	2	1	1	1	1	1	0	2	1	1	7	1
7	1	2	1	1	1	1	1	7	2	1	1	7	1
7	1	2	1	1	1	1	1	7	2	3	1	7	1
7	3	2	1	1	1	1	1	7	2	3	1	7	1
7	3	2	1	3	1	1	1	7	2	3	1	7	1
7	3	2	3	3	1	1	1	7	2	3	1	7	1
7	3	3	3	3	1	1	1	7	2	3	1	7	1
7	3	3	3	3	1	1	1	7	2	3	4	7	1
7	3	3	3	3	1	1	4	7	2	3	4	7	1
7	3	3	3	3	1	4	4	7	2	3	4	7	1
7	3	3	3	3	4	4	4	7	2	3	4	7	1
7	3	3	3	3	4	4	4	7	4	3	4	7	1
7	3	3	3	3	4	4	4	7	4	3	4	7	2
7	3	3	3	3	4	4	4	7	4	5	4	7	2
7	5	3	3	3	4	4	4	7	4	5	4	7	2
7	5	3	3	5	4	4	4	7	4	5	4	7	2
7	5	3	5	5	4	4	4	7	4	5	4	7	2
7	5	5	5	5	4	4	4	7	4	5	4	7	2
7	5	5	5	5	4	4	6	7	4	5	4	7	2
7	5	5	5	5	4	4	6	7	4	5	4	7	3
7	5	5	5	5	4	4	6	7	4	5	6	7	3
7	5	5	5	5	6	4	6	7	4	5	6	7	3
7	5	5	5	5	6	4	6	7	6	5	6	7	3
7	5	5	5	5	6	6	6	7	6	5	6	7	3
7	5	5	5	5	6	6	6	7	6	5	6	7	4

Verificamos que o guarda redes quando não forma equipa passa do estado WAITING_TEAM (1) para o WAITING_START_1/ WAITING_START_2 (3/4), não passando pelo estado FORMING_TEAM (2).

P00	P01	P02	P03	P04	P05	P06	P07	P08	P09	G00	G01	G02	R01
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1	0	0	0	0	0	0
1	1	0	1	0	0	1	1	0	0	0	0	0	0
1	1	1	1	0	0	1	1	0	0	0	0	0	0
1	1	1	1	1	0	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1	7	0	0	0	0	0
1	1	1	1	1	1	1	1	7	0	0	0	0	0
1	1	1	1	1	1	1	1	7	0	2	0	0	0
1	1	1	1	1	1	1	1	7	0	2	0	0	0
1	1	1	1	1	1	1	1	7	0	2	0	0	0
1	1	1	1	1	1	1	1	7	7	2	0	0	0
1	3	1	1	1	1	1	1	7	7	2	0	0	0
1	3	1	1	1	1	1	1	7	7	2	0	2	0
1	3	1	1	1	1	1	1	7	7	2	0	2	0
1	3	1	3	1	1	1	1	7	7	2	0	2	0
1	3	1	3	1	1	1	1	7	7	2	7	2	0
3	3	1	3	1	1	1	1	7	7	2	7	2	0
3	3	1	3	1	1	1	3	7	7	2	7	2	0
3	3	1	3	1	1	1	3	7	7	3	7	2	0
3	3	1	3	1	1	4	3	7	7	3	7	2	0
3	3	1	3	1	1	4	3	7	7	3	7	2	1
3	3	4	3	1	1	4	3	7	7	3	7	2	1
3	3	4	3	4	1	4	3	7	7	3	7	2	1
3	3	4	3	4	5	4	3	7	7	3	7	2	1
3	3	4	3	4	5	4	3	7	7	3	7	4	1
3	3	4	3	4	5	4	3	7	7	3	7	4	2
3	5	4	3	4	5	4	3	7	7	3	7	4	2
3	5	4	5	4	5	4	3	7	7	3	7	4	2
5	5	4	5	4	5	4	3	7	7	3	7	4	2
5	5	4	5	4	5	4	3	7	7	5	7	4	2
5	5	4	5	4	5	6	3	7	7	5	7	4	2
5	5	4	5	6	5	6	3	7	7	5	7	4	2
5	5	4	5	6	5	6	3	7	7	5	7	4	3
5	5	4	5	6	7	6	3	7	7	5	7	4	3
5	5	4	5	6	7	6	3	7	7	5	7	6	3
5	5	4	5	6	7	6	5	7	7	5	7	6	3
5	5	6	5	6	7	6	5	7	7	5	7	6	3
5	5	6	5	6	7	6	5	7	7	5	7	6	4

Neste teste, verificamos que, quando chega e pode formar equipa, nunca assume o estado WAINTING_TEAM (1) e passa logo para o estado FORMING_TEAM (2).

Teste do Player

P00	P01	P02	P03	P04	P05	P06	P07	P08	P09	G00	G01	G02	R01
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	0	0	0	0	0	0	0	0	0
1	0	1	1	0	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	1	0	0	0	0	0
1	1	1	1	1	1	0	0	1	0	0	0	0	0
1	1	1	1	1	1	0	0	1	0	0	0	0	0
1	1	1	1	1	1	0	0	1	0	0	0	0	0
1	1	1	1	1	1	0	0	1	0	0	0	0	0
1	1	1	1	1	1	0	0	1	0	2	0	0	0
1	1	1	1	1	1	1	0	1	0	2	0	0	0
1	1	1	1	1	1	1	0	1	0	2	0	2	0
1	1	1	1	1	1	1	0	1	0	2	0	2	0
1	1	1	1	1	1	1	0	1	7	2	0	2	0
1	1	1	1	1	1	1	0	1	7	2	0	2	1
1	1	3	1	1	1	1	0	1	7	2	0	2	1
1	1	3	3	1	1	1	0	1	7	2	0	2	1
3	1	3	3	1	1	1	0	1	7	2	0	2	1
3	1	3	3	1	3	1	0	1	7	2	0	2	1
3	1	3	3	1	3	1	0	1	7	3	0	2	1
3	1	3	3	4	3	1	0	1	7	3	0	2	1
3	1	3	3	4	3	1	0	4	7	3	0	2	1
3	1	3	3	4	3	1	7	4	7	3	0	2	1
3	4	3	3	4	3	1	7	4	7	3	0	2	1
3	4	3	3	4	3	4	7	4	7	3	0	2	1
3	4	3	3	4	3	4	7	4	7	3	0	4	1
3	4	3	3	4	3	4	7	4	7	3	0	4	2
3	4	3	3	4	3	4	7	4	7	3	7	4	2
3	4	3	5	4	3	4	7	4	7	3	7	4	2
3	4	5	5	4	3	4	7	4	7	3	7	4	2
5	4	5	5	4	3	4	7	4	7	3	7	4	2
5	4	5	5	6	3	4	7	4	7	3	7	4	2
5	6	5	5	6	3	4	7	4	7	3	7	4	2
5	6	5	5	6	3	4	7	4	7	3	7	4	3
5	6	5	5	6	3	4	7	6	7	3	7	4	3
5	6	5	5	6	3	6	7	6	7	3	7	4	3
5	6	5	5	6	3	6	7	6	7	5	7	4	3
5	6	5	5	6	3	6	7	6	7	5	7	6	3
5	6	5	5	6	5	6	7	6	7	5	7	6	3
5	6	5	5	6	5	6	7	6	7	5	7	6	4

Verificamos que trocar de estado para a equipa correspondente WAITING_START (3/4), e quando o árbitro troca para o estado FORMING_TEAM (2), os jogadores vão começar a trocar progressivamente para o PLAYING (5/6), dependendo da equipa. Verificamos, ainda, que, quando o árbitro assume o estado REFEREEING (3), os jogadores já estão todos nos devidos estados.

P00	P01	P02	P03	P04	P05	P06	P07	P08	P09	G00	G01	G02	R01
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0	0	0	0	0	0	0
1	0	1	1	1	1	1	0	0	0	0	2	0	0
1	0	1	1	1	1	1	0	0	0	0	2	1	0
1	0	1	1	1	1	1	0	1	0	0	2	1	0
1	0	1	1	1	1	1	0	1	0	7	2	1	0
1	0	1	1	1	1	1	0	1	0	7	2	1	1
1	0	1	1	1	1	1	0	1	2	7	2	1	1
1	0	1	1	1	1	1	7	1	2	7	2	1	1
1	7	1	1	1	1	1	7	1	2	7	2	1	1
1	7	1	1	3	1	1	7	1	2	7	2	1	1
1	7	3	1	3	1	1	7	1	2	7	2	1	1
1	7	3	3	3	1	1	7	1	2	7	2	1	1
3	7	3	3	3	1	1	7	1	2	7	2	1	1
3	7	3	3	3	1	1	7	1	2	7	3	1	1
3	7	3	3	3	4	1	7	1	2	7	3	1	1
3	7	3	3	3	4	4	7	1	2	7	3	1	1
3	7	3	3	3	4	4	7	4	2	7	3	1	1
3	7	3	3	3	4	4	7	4	2	7	3	4	1
3	7	3	3	3	4	4	7	4	4	7	3	4	1
3	7	3	3	3	4	4	7	4	4	7	3	4	2
3	7	3	3	5	4	4	7	4	4	7	3	4	2
3	7	5	3	5	4	4	7	4	4	7	3	4	2
5	7	5	3	5	4	4	7	4	4	7	3	4	2
5	7	5	5	5	4	4	7	4	4	7	3	4	2
5	7	5	5	5	4	4	7	4	4	7	5	4	2
5	7	5	5	5	6	4	7	4	4	7	5	4	2
5	7	5	5	5	6	6	7	4	4	7	5	4	2
5	7	5	5	5	6	6	7	6	4	7	5	4	2
5	7	5	5	5	6	6	7	6	4	7	5	4	3
5	7	5	5	5	6	6	7	6	4	7	5	6	3
5	7	5	5	5	6	6	7	6	6	7	5	6	3
5	7	5	5	5	6	6	7	6	6	7	5	6	4

Observamos que, quando o jogador chega e já existem jogadores e guarda redes disponíveis para formar equipa, este não assume o estado WAITING_TEAM (1) passando, e bem, logo para o estado FORMING_TEAM (2) para formar a equipa.