**Training**

1. **Pseudo-code**

// Start of the training

for i=0 to 900

// FIELD’S PART --------

// receive players’ initial positions

MPI\_Recv(&initial\_players\_positions[], 2, MPI\_INT, p+1, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

// send current ball position to players

MPI\_Send(ball\_position, 2, MPI\_INT, p+1, 0, MPI\_COMM\_WORLD);

// PLAYERS’ PART --------

// send my position to the field

MPI\_Send(my\_position, 2, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

// receive the ball position

MPI\_Recv(ball\_position, 2, MPI\_INT, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

for j=0 to NUM\_STEPS (10 steps overall)

// FIELD’S PART --------

// receive indices of players that reached the ball during a round

MPI\_Recv(&playersReachedBall[p-1], 1, MPI\_C\_BOOL, p, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

// choose a winner if there are a several players reached the ball simultaneously

winnerIndex = who\_is\_winner(playersReachedBall ,numOfPlayers); // if '-1', there is no winner

// send the winner index to each player so they can detect who won the ball

MPI\_Send(&winner, 1, MPI\_C\_BOOL, p+1, 0, MPI\_COMM\_WORLD);

// since the field knows the winner index, it receives a new position of the ball right from

// that player

MPI\_Recv(new\_ball\_position, 2, MPI\_INT, winnerIndex+1, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

// receive all the data that should go to output after each round: total number of meters

// each player ran, times each of them reached the ball, times each of them won the ball,

// new players positions

MPI\_Recv(&players\_meters[p], 1, MPI\_INT, p+1, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

MPI\_Recv(&times\_reached\_ball[p], 1, MPI\_INT, p+1, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

MPI\_Recv(&times\_won\_ball[p], 1, MPI\_INT, p+1, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

MPI\_Recv(&players\_positions[p\*2], 2, MPI\_INT, p+1, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

// PLAYERS’ PART --------

// if I reached the ball, increase total number of meters I ran

if ((gotBall = did\_reach\_ball(my\_position, ball\_position))) {

timesReachedBall++;

}

// Send the data showing wether I got the ball in this round or not

MPI\_Send(&gotBall, 1, MPI\_C\_BOOL, 0, 0, MPI\_COMM\_WORLD);

// receive the winner index

MPI\_Recv(&winner, 1, MPI\_C\_BOOL, 0, 0, MPI\_COMM\_WORLD, MPI\_STATUS\_IGNORE);

// if I’m a winner, I kick the ball and send a new ball position to the field

if (winner) {

kick\_ball(ball\_position, new\_ball\_position);

MPI\_Send(new\_ball\_position, 2, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

// send all the data that should go to output after each round: total number of meters

// I ran, times I reached the ball, times I won the ball,

// my new position

MPI\_Send(&metersTotal, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

MPI\_Send(&timesReachedBall, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

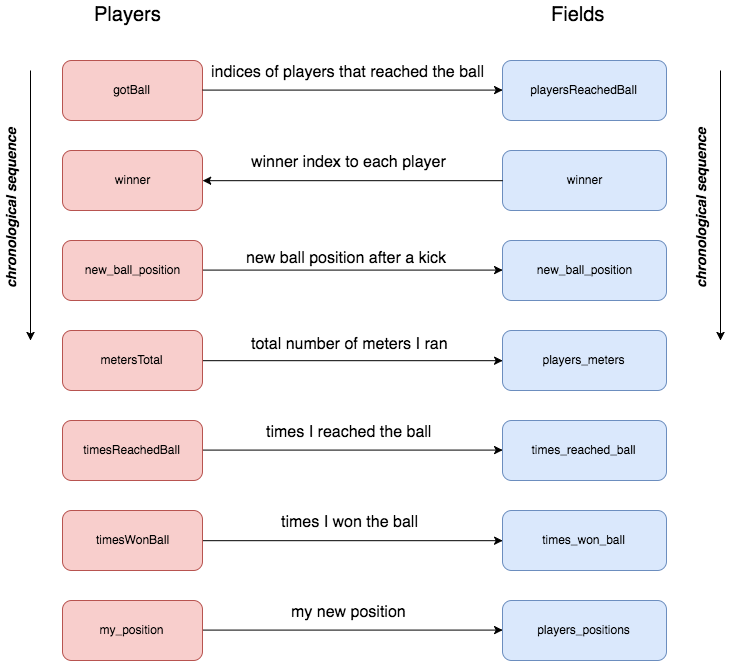
MPI\_Send(&timesWonBall, 1, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

MPI\_Send(my\_position, 2, MPI\_INT, 0, 0, MPI\_COMM\_WORLD);

}

} // End of the game

1. **Communication diagram**

****

**Match**

1. **Pseudo-code**

// initializing 4 communicators (teamAF and teamBF – communicators for teamA and fields and // teamB and fields respectively)

MPI\_Comm\_create(teamA);

MPI\_Comm\_create(teamB);

MPI\_Comm\_create(teamAF);

MPI\_Comm\_create(teamBF);

// each process is assigned three parameters on a random basis

assign\_skills(speed, dribbling, kickPower, rank)

// each field is assigned with its borders on a football field

assign\_borders(myBorders, rank)

// Start of the game

// two halves of the game time

for j=0 to 1 {

//each player is assigned with the position on a football field - each team starts on its //own half of the field

assign\_position(myPosition, myFieldSector, rankA, checkTeam, j);

// the ball is positioned at the center

ballPosition[0] = ROW\_BALL\_INIT\_POS;

ballPosition[1] = COL\_BALL\_INIT\_POS;

//2700 rounds in each half

for i=0 to 2700 {

// fields gather initial positions of players in order to output this info in console;

// two different communicators are used for getting positions from both teams;

// thereafter we use both communicators so as to collect/broadcast info from/to

// players of different teams (they are shortened to A/B);

// instead of using both

// MPI\_Gather(myPosition, 2, MPI\_INT, initialPlayersPositionsA, 2, MPI\_INT, NUM\_Players, teamAandFields\_COMM);

// MPI\_Gather(myPosition, 2, MPI\_INT, initialPlayersPositionsB, 2, MPI\_INT, NUM\_Players, teamBandFields\_COMM);

// the shortened version is written

MPI\_Gather(myPosition, 2, MPI\_INT, initialPlayersPositionsA/B, 2, MPI\_INT, NUM\_Players, teamA/BandFields\_COMM);

// knowing a ball position, each field checks if the ball is in its borders;

// IGotBall then is used by players of both teams to get the rank of the process

// possessing the ball

IGotBall = is\_ball\_in\_my\_borders(myBorders, ballPosition);

// players collect info on who possesses the ball into ballPossessionIndex array

MPI\_Allgather(&IGotBall, 1, MPI\_C\_BOOL, ballPossessionIndex, 1, MPI\_C\_BOOL, teamA/BandFields\_COMM);

// players receive the rank of a field with the ball using info in

// ballPossessionIndex

fieldWithBallIndex = get\_field\_index\_with\_ball(ballPossessionIndex);

// having 'fieldWithBallIndex' received, players know who they collect info on

// 'ballPosition' from;

// thus, players receive 'ballPosition'

MPI\_Bcast(ballPosition, 2, MPI\_INT, fieldWithBallIndex, teamA/BandFields\_COMM);

// FP0 (NUM\_Players) collect players' positions in order to broadcast it afterwards to players

MPI\_Gather(myPosition, 2, MPI\_INT, allPlayersPositions\_A/B, 2, MPI\_INT, NUM\_Players, teamA/BandFields\_COMM);

// players collect positions of teammates from FP0 (NUM\_Players)

MPI\_Bcast(allPlayersPositions\_A/B, 2, MPI\_INT, NUM\_Players, teamA/BandFields\_COMM);

// players take decision on what they do next - move or stay on the field till the next round

make\_step(myPosition, myFieldSector, speed, ballPosition);

// FP0 collects positions of all players in 'allPlayersPositionsA/B' arrays, so it can choose a

// winner

MPI\_Gather(myPosition, 2, MPI\_INT, allPlayersPositions\_A/B, 2, MPI\_INT, NUM\_Players, teamA/BandFields\_COMM);

// each player calculates a ballChallenge; the value will be used in future in case the player // competes with an opponent

ballChallenge = ((rand() % 10) + 1) \* dribbling;

// FP0 gathers all the ballChallenges of players

MPI\_Gather(&ballChallenge, 1, MPI\_INT, ballChallengeA/B, 1, MPI\_INT, NUM\_Players, teamA/BandFields\_COMM);

// FP0 identifies the index and the team of a winner

identify\_winner(allPlayersPositions\_A, allPlayersPositions\_B, ballPosition, winner, reachedBall\_A, reachedBall\_B);

// if there are two competitors for the ball, the winner is being chosen on a random basis

choose\_winner(ballChallengeA, ballChallengeB, winner, wonBall\_A, wonBall\_B);

// send 'winners' arrays so that each player can detect if he is a winner

MPI\_Bcast(winner, NUM\_Players, MPI\_C\_BOOL, NUM\_Players, teamA/BandFields\_COMM);

// if a player wins the ball, he kick it to a certain position according to the strategy

if ((didWinBall = did\_win\_ball(winner, rankA, checkTeam)))

kick\_ball(ballPosition, myPosition, myFieldSector, allPlayersPositions\_A, goalXcoord, newBallPosition, kickPower);

// FP0 receives new ball position (or a previous position if there were no winners) and sends // it to players

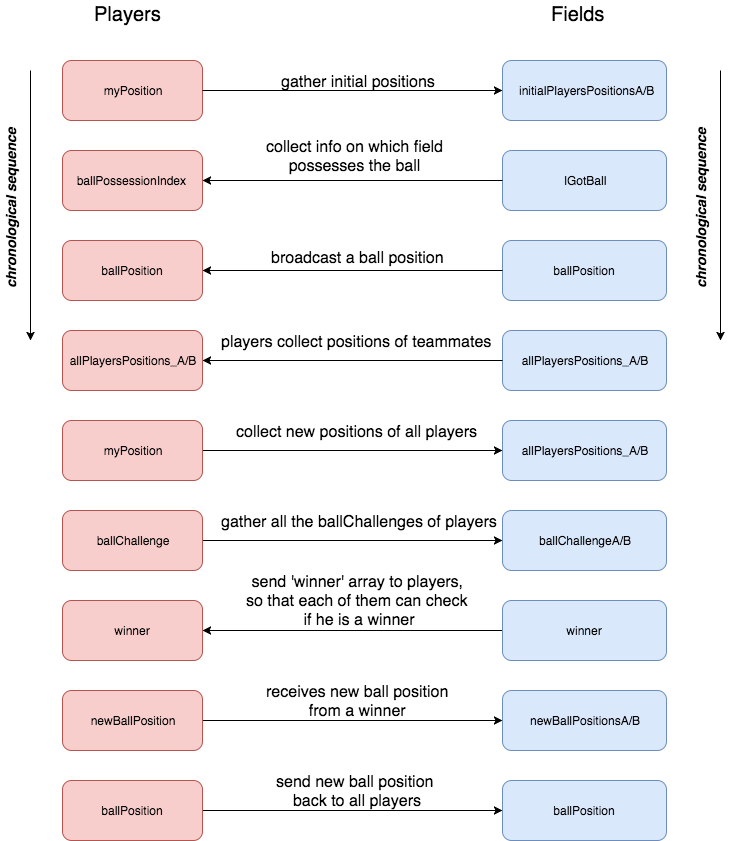
MPI\_Gather(newBallPosition, 2, MPI\_INT, newBallPositionsA/B, 2, MPI\_INT, NUM\_Players, teamA/BandFields\_COMM);

MPI\_Bcast(ballPosition, 2, MPI\_INT, NUM\_Players \* 2, MPI\_COMM\_WORLD);

} // end of a round

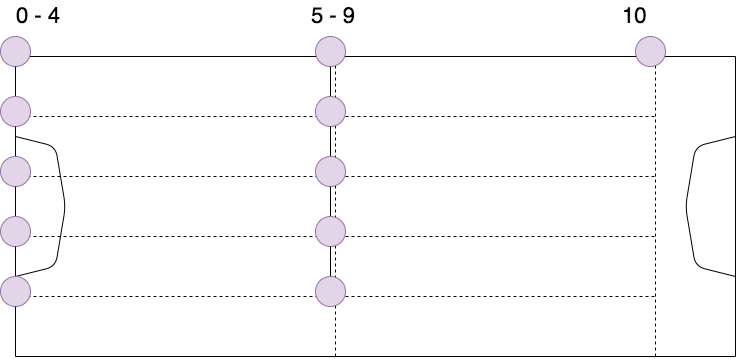
} // end of the game

1. **Communication diagram**

****

1. **Walkthrough**

**Players positions and strategy**

****

The example scheme from above illustrates initial positions of players of Team A in a first half, which seeks to score a goal at the right goalpost. Players are marked with purple circles. Each player is only allowed to move inside its borders - dash lines, it prevents a game from being a mess since each player takes only its own space. Once a player got the ball, his strategy would be as follows:

* calculate the distance to a goalpost
* calculate the distances to all teammates between me and a goalpost
* if there are some players that are closer to me than a goalpost – choose the closest one and give him a pass (it might take several kicks as the kick distance is restricted by rules); if the goalpost is closer to you than any of your teammates, kick the ball towards the goalpost

In the next half of the game, the initial positions and a target goalpost will be symmetrically reflected. The same is for Team B.