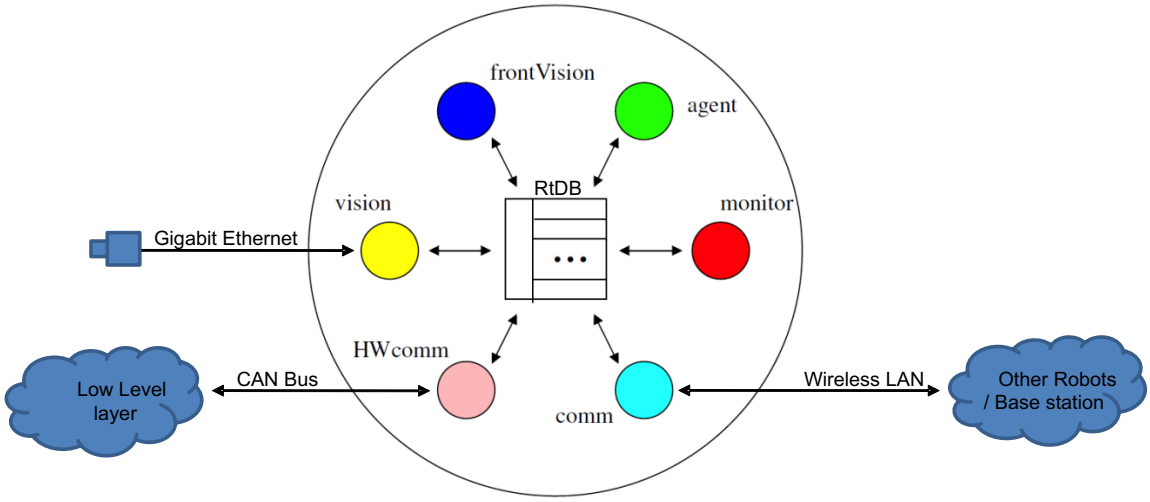
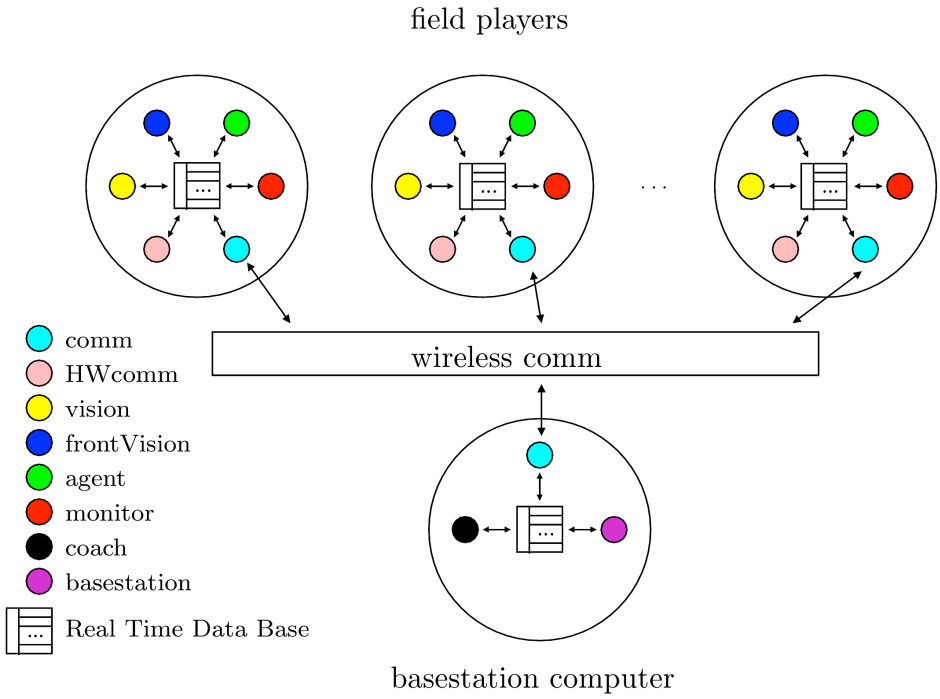
CAMBADA software architecture

1. High-level layer



2. Overall software architecture



The Realtime Data Base (RtDB) is a middleware that provides a seamless access to the complete team state using a distributed database, partially replicated to all team members.

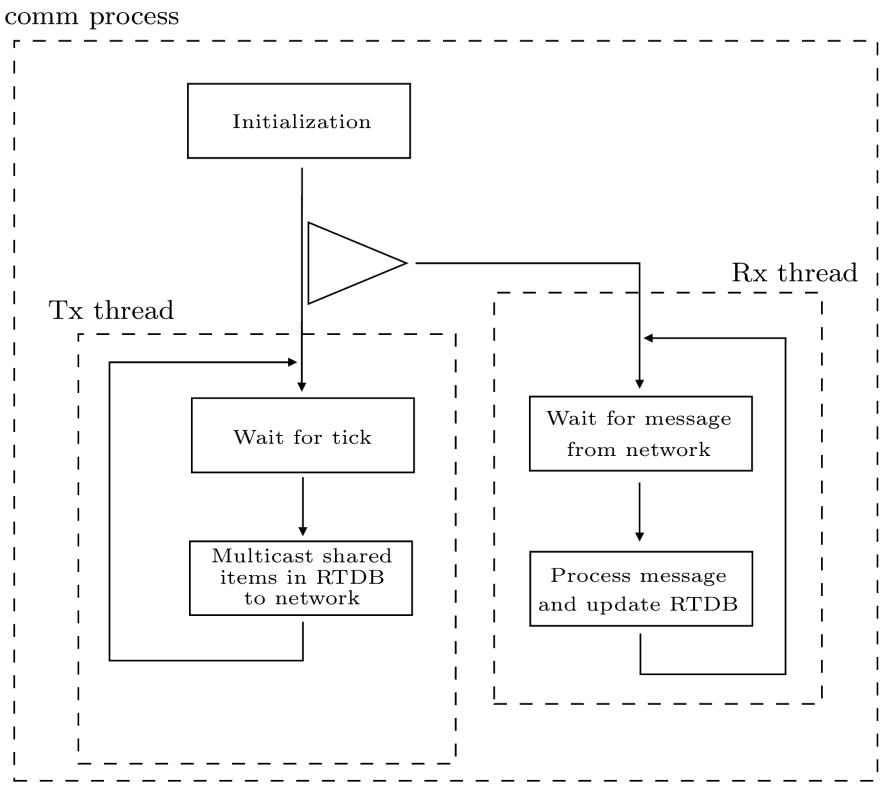
The information in the RtDB includes, for example, the absolute positions and postures of all players, as well as the position of the ball in global coordinates. The structure also encompasses some coordination variables, used for the regulation of teamwork tasks.

The RtDB is also intensively used as an inter-process communication mean, due to its capability of defining local memory items that are not broadcasted to other robots, but are still accessible by other processes running on the same agent environment. This local information includes sensorial data from the vision system and the hardware platform and also commands that are sent to the low-level control layer through a gateway interface.

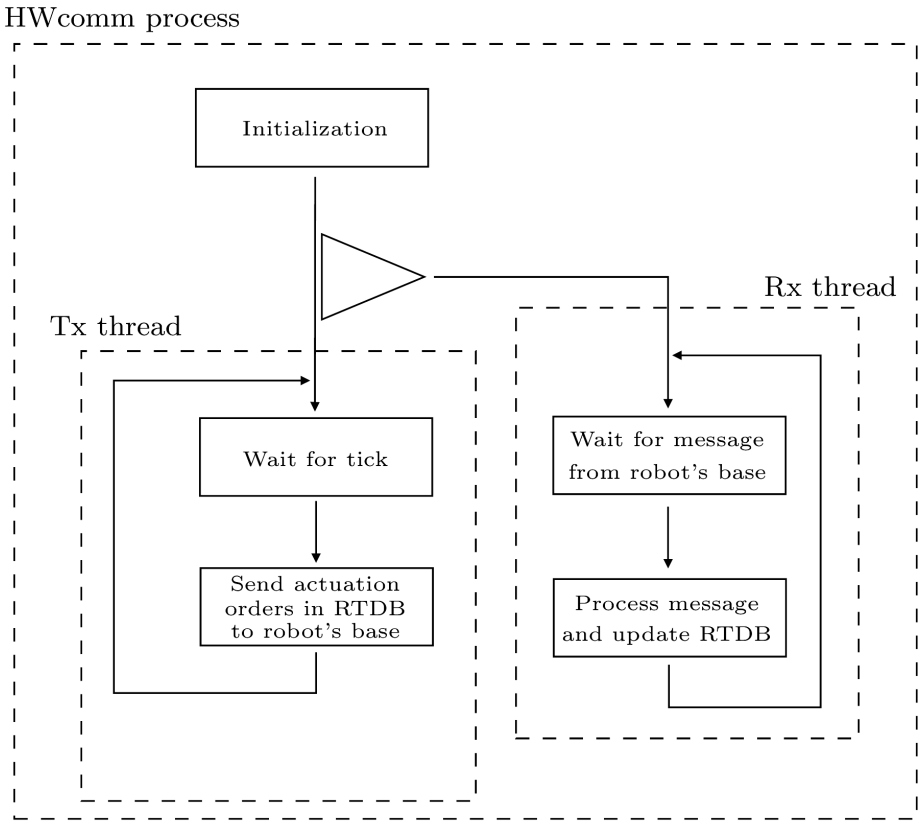
In a nutshell, the description of each process is the following:

- **monitor**
  - Controls the other processes running on the same PC. If a process fails, the monitor re-launches it.
- **comm**
  - This process handles the Wi-Fi communication both for sending information to the multicast group and receiving data from other agents.
- **HWcomm**
  - Used to communicate with the low-level layer via a USB/CAN gateway.
- **vision**
  - Process responsible for capturing a frame from the digital camera and analyzing it, extracting the relevant information. That data is then stored at a local RtDB item for the agent to use.
- **frontVision**
  - Similar to vision, but handles a different camera, mounted in the front of the robot.
- **agent**
  - The process responsible for decision, coordination and reasoning.
- **basestation**
  - An application used to visualize and control the robots state remotely.
- **coach**
  - A software agent responsible for controlling the team basic strategy. It defines formation and assigns robots to specific strategic positions.

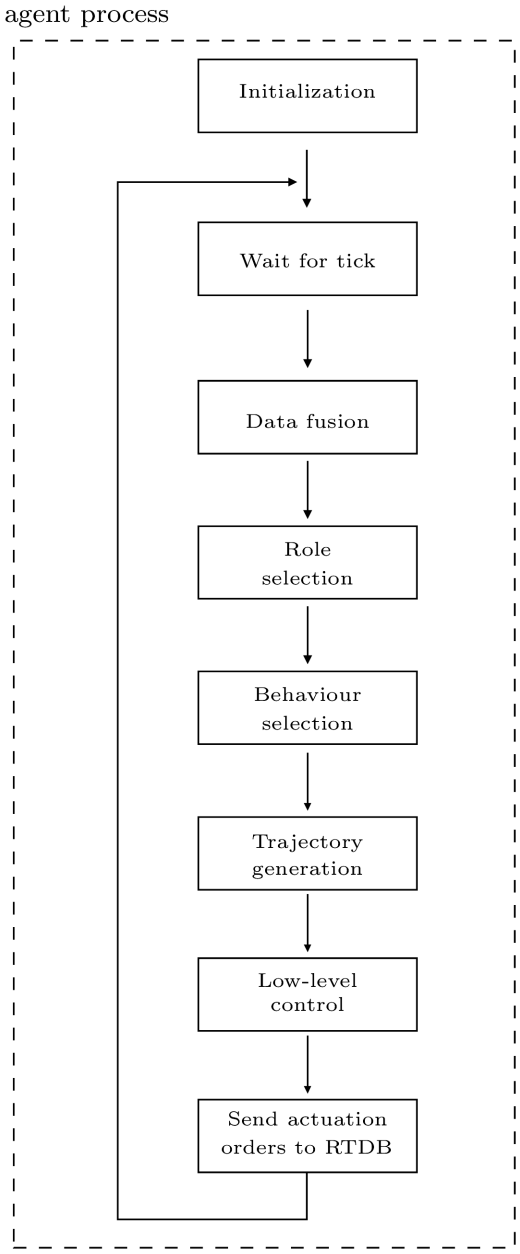
3. comm process



4. `HWcomm` process



5. `agent` process



5.1. Decision and Coordination

The `Roles` and `Behaviours` are constructed and initialized in the “initialization” stage and may be selected in the loop. This allows these blocks to hold a persistent internal state throughout the agent lifespan, avoiding problems regarding history loss.

The `Role` selection is performed using a Finite State Machine (FSM), using conditional statements. Then, the selected role uses an `Arbitrator` that contains a set of `Behaviors` and picks one at runtime using a set of rules, as an alternative to FSM. These rules can be simple condition statements, more complex utility function evaluations or even a mixture of both.

The `Behaviour` then uses a `Controller` to generate target velocities and may define that the robot should perform a lob-shot, a pass, set the grabber mode, etc.

6. `vision` process