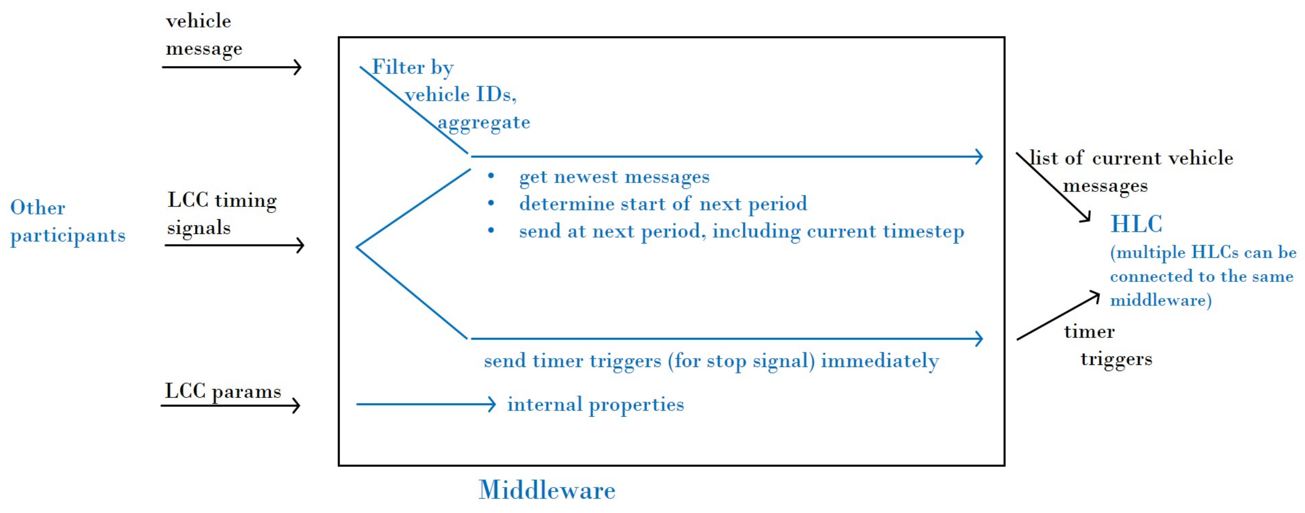
Middleware Structure

Middleware Workflow

Shared or common functionality for the communication via DDS is bundled in the *cpm* library (*cpm_lib*). This library is *written in C++* and is used by the vehicle, the LCC and the Middleware. The HLC, on the other hand, is supposed to be more or less *independent from the implementation language* of the library. Programmers of the HLC programs should not be troubled with its implementation details. Thus, it was decided to create a Middleware which performs most of the communication tasks that otherwise would have to be implemented in some form in the HLC, which might not be able to use a C++ library.

The Middleware *coordinates the communication* with the DDS domain of the vehicle, LCC and further participants with the HLC. It is responsible for *sendin g timed pre-aggregated vehicle information* to the HLC. These can be interpreted as a start signal for starting the computation using this data. It *sends the commands generated by the HLC* as a reaction to this start signal to the vehicles. Any *timing* task is performed by the Middleware, so period and offset for the interaction with the HLC are set here (and can be set when starting the application using command line parameters). The HLC is *informed about the current time*, in case that is necessary for its own computations.



vehicle message
Other participants
LCC timing signals
LCC params
Filter by vehicle IDs, aggregate
get newest messages
determine start of next period
send at next period, including current timestep
send timer triggers (for stop signal) immediately
internal properties
Middleware
list of current vehicle messages
HLC
(multiple HLCs can be connected to the same middleware)
timer triggers

Figure: Communication from other participants to the HLC filtered by the Middleware

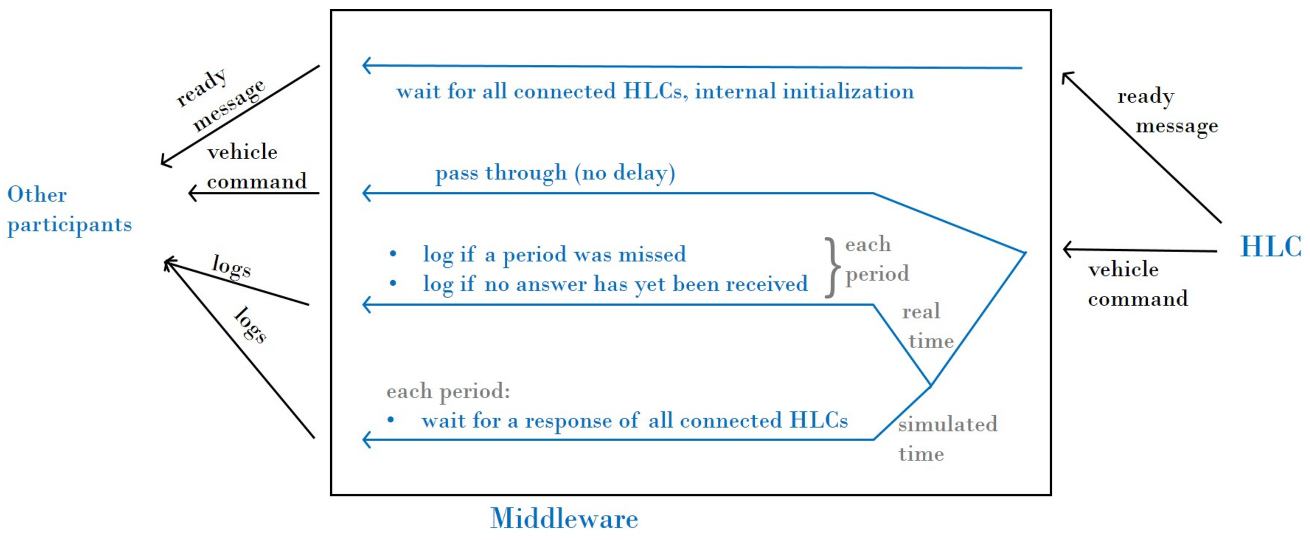


Figure: Communication from the HLC to other participants

The middleware also manages further internal timing according to the current simulated and real time, propagates the LLC's start and stop signals and waits for the HLC to finish its computation in the simulated case. It is also responsible for logging communication or timing errors.

The HLC only needs to integrate and configure a reader and a writer to listen to the Middleware and send commands. Thus, as long as the programming language is supported by RTI DDS, the programmer does not need to know the implementation details of the communication.

Functionality

- Logging
- Waiting for the HLC to join the domain & send a ready message
- Periodically calling the HLC to request commands for the vehicle
- Forwarding vehicle state information to the HLC periodically (always the newest information is sent) + timing information s.t. the HLC does not need to care about getting the right timestamps itself
- Forwarding commands sent by the HLC to the vehicle
- Forwarding stop signals to the HLC
- Receiving (some) parameters from the parameter server, e.g. the period with which to call the HLC script