PRACTICAL COURSE

FROM SENSORS TO DRIVING FUNCTIONS - DEVELOP YOUR OWN CAR

V2V Communication

To allow the vehicles to communicate among each other a WiFi network is used. One of the WiFi equipped Raspberry Pis in each vehicle is connected to a WiFi router that spans a wireless local network.

UDP Communication

To allow for a simple yet efficient exchange of messages among the vehicles on the common network, UDP was chosen for data transmission. All **V2V!** (**V2V!**) messages are transmitted as UDP Broadcasts and therefore can be received by any device connected to the network without further authentication. This ensures that every vehicle on the network can receive all relevant messages while a transmitting vehicle neither does have to know about the number and presence of particular vehicles nor it has to address them individually.

As the communication is to be carried out on Raspberry Pis running Linux, implementation was done using socket communication. The mechanism of sockets is provided by the kernel and allows for a certain level of abstraction. As soon as the socket was opened for the remaining part of the implementation it does not matter whether an IP based interface or some other communication interface is used for low-level message transport. For the current implementation this also means that it is not bound to one particular network interface but it can utilize arbitrary wired or wireless interfaces.

V2V Message Structure

Figure 1.1 illustrates the telegram structure used for communication among vehicles. The Messages for **V2V!** communication are transported in the payload data field of UDP packets. One UDP packet carries exactly one **V2V!** message.

The **V2V!** messages themselves are split into a header and a payload section.

The payload section of a **V2V!** message carries the actual **V2V!** message. The content of this section resembles the byte-wise representation of arbitrary message structures defined in the autoFOCUS 3 model. The structure and the meaning of the content of the payload data field remains unknown to the implementation of the **V2V!** message transport mechanism and therefore freely can be modified in the model without having to adapt the implementation of the communication layer.

The header section of a **V2V!** message is made up of the following parts:

- The **UID!** (**UID!**) of the message transmitter

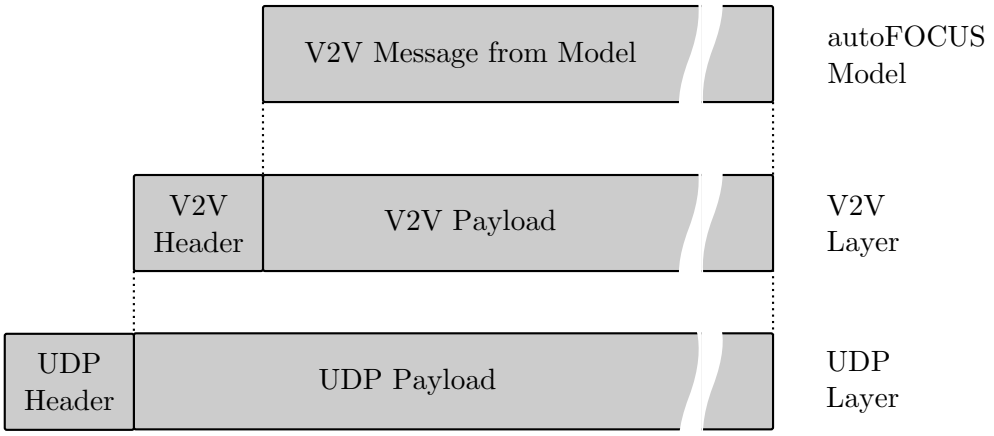

Figure 1.1: **V2V!** Communication Telegram Structure

- The message type identifier
- The message payload length

The transmitter **UID!** allows a receiver of this message to determine from which vehicle the message originated. This information is used for message filtering as described in section

The message type identifier uniquely associates the message to one of currently 8 predefined message types. This information is used to allow the appropriate handling of the **V2V!** message payload field content — e.g. in the current implementation this information is used to forward the message to the right message buffer to be accessible form the appropriate input port of an autoFOCUS 3 model.

The message payload length field gives information on how many bytes of actual **V2V!** message are carried in the payload data field. As the structure of the actual **V2V!** messages is unknown to the message transport implementation so is their length. Therefore the implementation uses the information form the message payload length field to handle the correct amount of bytes.

Message Filtering

The concept of sending **V2V!** messages as broadcasts initially means that every vehicle on the network receives all messages transmitted assuming an error free communication. However not all **V2V!** messages received by one vehicle are actually relevant for this particular vehicle.

Firstly by receiving all messages on the network a vehicle also will receive messages that were sent by this particular vehicle itself. Obviously such messages should not be processed locally but their content safely can be discarded.

Secondly there are some message types that are only of interest for a vehicle if they originate form one particular other vehicle. At the current state of the project this includes messages form the platoon leader that are only relevant if the local vehicle actually is part of the platoon the message originated form as well as messages that originate from the vehicle directly preceding the local vehicle and are only relevant for the local vehicle.

To achieve that only messages that are relevant for a particular vehicle are processed a message filtering mechanism was implemented.

As every message carries the **UID!** of the vehicle that it was transmitted from in the header it is easy to judge on the relevance of a received message: If the **UID!** in the message is the same as the **UID!** of the local vehicle, the received message originated from the local vehicle and can be safely ignored. If the **UID!** in the message is not the same as the local **UID!** the content of the message header is processed further to determine the type of the message. If the message is of a type that is only relevant if the message originates from one particular vehicle, message processing is aborted if the **UID!** in the message header does not equal the **UID!** of the source vehicle of interest. If non of the above mentioned points is fulfilled the message is processed and the message payload is eventually made available to the autoFOCUS 3 model.

As communication currently is done via WiFi every vehicle has an IP address that is unique in the network and therefore could have been used to determine the origin of a message for filtering. While using the IP addresses at first may sound like the more intuitive way there are some reasons against it that give reason for rather using **UID!**s:

First of all the current approach uses WiFi as it is a readily available and easy to use technology well suited for experiments at the current scale of the project. Still it cannot be taken for granted that WiFi will be the communication path of choice when findings of this project are to be transferred to real cars or trucks. As real world solutions might use entirely different communication channels it does not seem expedient to rely on the fact that there always will be something like an IP address usable for message filtering.

Furthermore from an implementation point of view accessing the IP address of a particular network adapter would contradict the level of abstraction accomplished by utilizing sockets for communication. In contrast for an **UID!** based filtering solution it does not matter which particular network interface is used, which especially is advantageous if there are multiple network interfaces present in one system as this is the case for most Linux computers.

Implementation

The communication mechanisms described above were implemented in C for Raspberry Pis running Linux.

The transmission of messages is done directly from the main thread of the application. The reception and initial handling of messages, however, is done in a separate thread running asynchronously to the main thread. This allows to always receive incoming messages, no matter which operations are currently executed in the main thread.

For each message type there is a buffer in memory. The receiver thread carries out the filtering procedure described in section and stores relevant incoming messages to the appropriate buffers. The main thread checks the content of these message buffers once per cycle and collects the stored messages for usage in the model.