HYCON2 MEETING REPORT

INRIA GRENOBLE

9th -12th January 2012 -Grenoble, France

Necs Team

January 17, 2011

The second Hycon2 workshop meeting was organized by INRIA-Grenoble. The aim for this work week was to continue the development of Aimsun/Matlab interface started in the first Hycon2 workshop in may 2011.

The group was composed by Alberto Nai Oleari, PHD student at University of Pavia, Jose Ramon Dominguez PHD student at University of Seville, Noortje Groot PHD student at Delft University of Technology, Mohammadreza Hajiahmadi PHD student at Delft University of Technology, Wenjie Lu PHD student at University of Grenoble, Dominik Pisarski PHD student at University of Grenoble, Luis Leon PHD student at University of Grenoble and Iker Bellicot Software Developer Engineer INRIA.

As said above, the aim was the culmination of the functional interface between the micro-simulator and Matlab. The scheme of the interconnections of these programs with the interface needed is shown in figure 1.

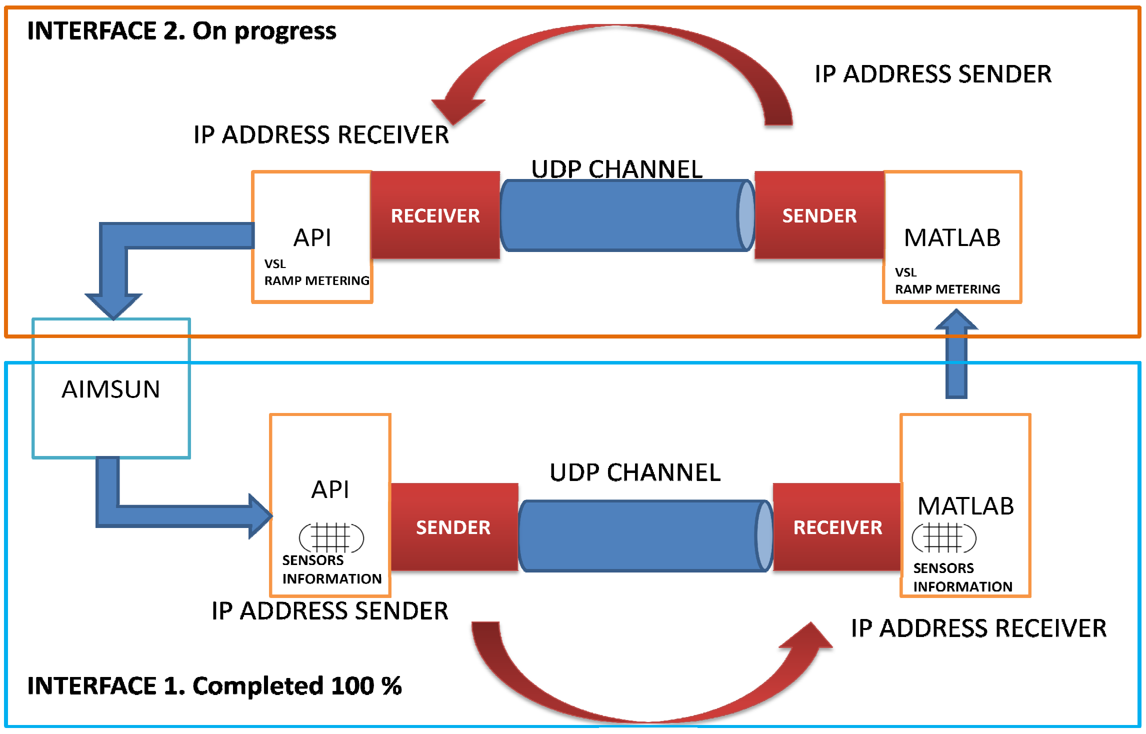


Figure 1. Scheme of the Interface between AIMSUM and MATLAB/SIMULINK

In order to finish the Interface 1 (Aimsun to Matlab), the work was divided into 2 steps:

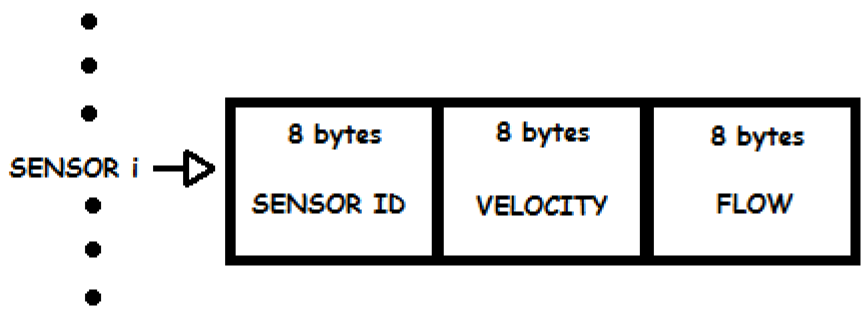
a) For this part, our goal was to generalize the API's C++ code in order to read all the sensors' information from Aimsum, as well as the data concerning sensors ID, the simulation time and the total period of simulation. Doing so, an array characterizing the situation on the highway every Time step during a fixed period of time was ready to be sent to Matlab. We used for this; the functions defined in Aimsun and presented in its tutorial.

b) In this step, the UDP communication needed to be established. In the previous workshop, we were able to send data of any sensor through the network in a general IP address and sniff these packets using the software Wireshark. Our goal this time was to send the array defined above to Matlab. To achieve so, a script .m on Matlab was created. This script, behaving as the receiver, allowed the UDP communication to begin.

   To establish this sender/receiver communication the function 'udp' on Matlab was applied. This function creates a UDP object associated with local host, remote port value, size of buffer, etc. Two problems were encountered during this process. The first was the default buffer size of Matlab's UDP packets (512 bytes), causing a huge problem of limitation regarding the information to be sent to the receiver.

   For each sensor the information requested into Matlab is: ID sensor (the sensor code on Aimsun), velocity and flow. These values are defined as doubles, size of 8 bytes. Therefore the array's size to be received is (24 x #sensors) bytes, quantity clearly bigger than the 512 bytes discussed before. This problem was overcome by defining the size of the buffer inside Matlab's UDP function 'udp' big enough to receive the whole datagram.

   A second issue came up when trying to send the packet:



   There was a disparity between the sender (C++) and receiver (Matlab) in the way to decode the data, causing a misreading of the packet. This problem was solved by introducing the right form and size of the packet inside the function fread (Read data from binary file) in Matlab.

Until this point the implementation of the interface 1 works in the following way:

Every Time step (time window) a datagram is prepared in the C++ API, this window can be chosen as small or as big as needed. This packet is sent to Matlab every Time step. The receiver stores velocity and flow values for its specific sensor. This process lasts as long as the complete time of the process. The values of the sensors can be used after the whole simulation on Aimsun is finished. A scheme depicting what was said before is showed in the Figure 2.

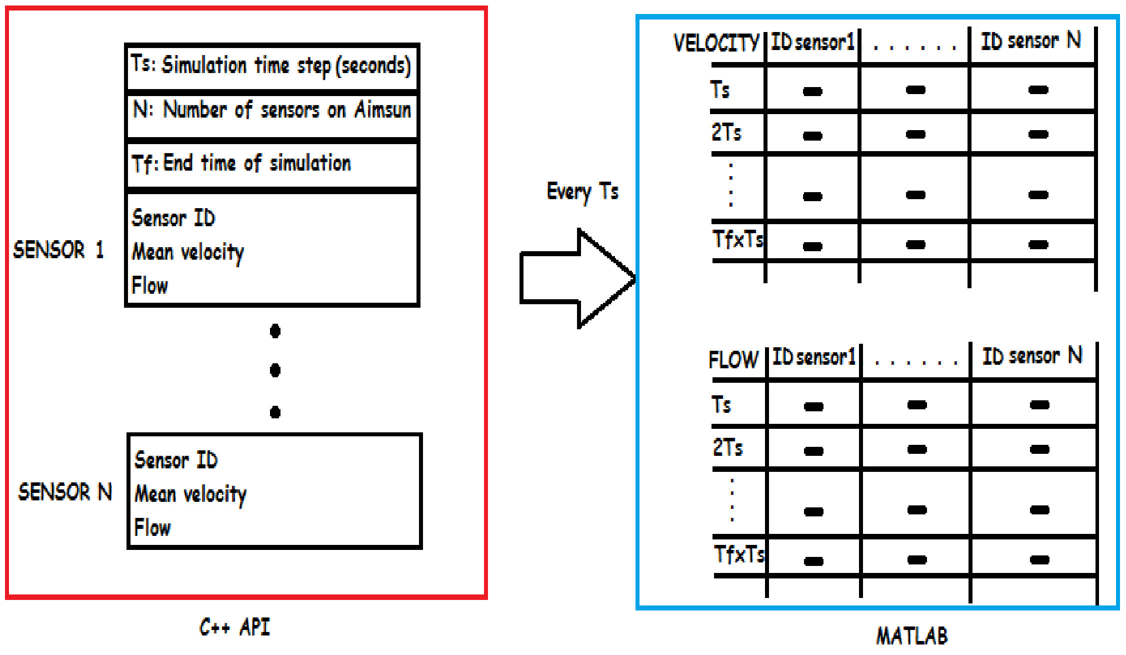


Figure 2. Scheme of communication AIMSUM and MATLAB

Continuing with this implementation, what needs to be added is the interface from Matlab to Simulink, in order to identify the different densities, flows and characteristic curves along the highway once all the sensors values are completely stored.

There is an important point that needs to be considered. If the control is going to be applied, then we need to be able to use every time step the information of velocity and flow of all the sensors. Hence, the Matlab code needs to be slightly modified with a cycle in order to capture every Time step the datagram and keep waiting for the next to arrive until a stop condition.

We might be inclined to conclude that the Interface 1 is complete, nevertheless depending on the scenario in which will be used; the Matlab code can be changed or improved.

Concerning the dual Interface (Matlab-Aimsun), we still have some work ahead of us to be done. Up this point we are able to change manually the Variable Speed Limit and Ramp metering inside the API. However, the control applied is to change these values through UDP protocol.

Therefore the interface 2 it’s still in progress and needs to be studied more in detail.