mapTransSock and mapRecvSock are used for remote visualization of ‘instances’. They allow for transmitting arbitrary data over a (TCP IP) network using the python socket library.

The normal (intended) transmitted packets are Map objects, which the receiving side (like remoteMapViewer.py) can directly use. In the intended structure of an ‘instance’ (as described in the ‘merged driverless instance’ document), the Map object is integrated (inherited) into the main ‘instance’ object. As a result, sending the whole ‘instance’ object could (with certain exceptions) work, but would be very inefficient, because it would also transmit function-class attributes and anything the user may have added manually. To avoid this (while maintaining Map transmission simplecity), the functions shallowCopyExtractMap() is used. The word ‘shallow’ refers to the fact that it only makes copies of pointers to objects, not copies of the objects, if a full (stand-alone) copy is needed, deepCopyExtractMap may be used. For map transmissions however, a shallow copy can speed up the copying process greatly, while transmitting the same (though potentially more volatile) data.

An important part of a remote-instance is the ability to affect the ‘instance’ through UI. This remote UI is handled through the UIparser() function in mapTransSock. These instructional packets are sent by a remote map viewer, which uses mapRecvSock and (usually) drawDriverless.py.

In (our implementation of) the python socket library, data can only be sent as (raw) bytes. For this reason (and possible compression benefits), the python pickle library is used. To transmit an object, it is pickled (converted to a bytes() object), sent, and (on the receiving end) unpickled (converted back into the original format).

The transmitted data can often become dispersed over several TCP packets. For this reason, a ‘packet header’ is introduced. The header is a string of constant length, which indicates the size of the packet.

Both mapTransSock and mapRecvSock have a function by the name of runOnThread(). This function is, as the name suggests, intended to be ran on a python Threading (library) thread.

An example of how to initialize the mapTransSock:

mapSender = MS.mapTransmitterSocket(host='', port=65432, objectWithMap=<instance>)

    mapSender.mapSendInterval = 0.2 #start safe, you can bring this number down if the connection is good (FPS = 1/this)

    threadKeepRunning = [True] #an argument (functional pointer) shared between the main and mapSockThread and main thread

    autoMapSend = [True] #an argument (functional pointer) shared between the main and mapSockThread and main thread

    UIreceive = [True] #an argument (functional pointer) shared between the main and mapSockThread and main thread

    mapSockThread = thr.Thread(target=mapSender.runOnThread, name="mapSockThread", args=(threadKeepRunning, autoMapSend, UIreceive), daemon=True)

    mapSockThread.start()

where <instance> is the ‘instance’ object.