Problem Description

The story should start with the last problem I had in last month, which is the memory overflow of the mote, because of the limited resources provided by Contiki. As the meeting we had just before your holiday, a good way to do that is to create a server from the node, and transmit the data to the server, and from the server we can create a Python file to read the data and do the complex computation via socket communication. At that meeting, you were very kind to show me the example code of Contiki for socket communication and the example code of Python for socket.

Then after that meeting, I found that actually the two examples cannot communicate with each other directly, I mean the socket example code of Contiki, and the socket example code of Python. Of course, I was using the same IP address and port for both sides. Then I though it might because the used port was not correct, so I tried a lot of port, as well as IP addresses. It turned out that they all failed.

Then I started to think another possibility that there might be something blocked in Linux so that Contiki cannot reach socket from Linux OS. Then I found a reference online [1], which is an exercise for Contiki and Cooja. In that document, a way to create an IPv6 network is introduced in Section 4.1. I found that the "*tunslip6*" tool might be the key point for my problem. "*tunslip6"* is the program that Contiki provided to set up interface on Linux IP stack and connect this interface via a socket to the node in Cooja, which is using port 60001 as default for the socket. So I used his method: I mounted a simple, light-weighted webserver on the node, in the code of this node, I transmit the personal lighting profile to the webserver, then start the "*tunslip6"* program. From the web browser, now I can see the profile is printed out on the screen. Then I created a Python file, to fetch this data from the webserver, and parse it to get the profile information, and then I transplant the complex computation of my conflict resolution algorithm onto this Python file, and finally I can see the result of this algorithm. Figure 1 is the slide that I was using to explain this setup in the meeting with Henk and Tanir.

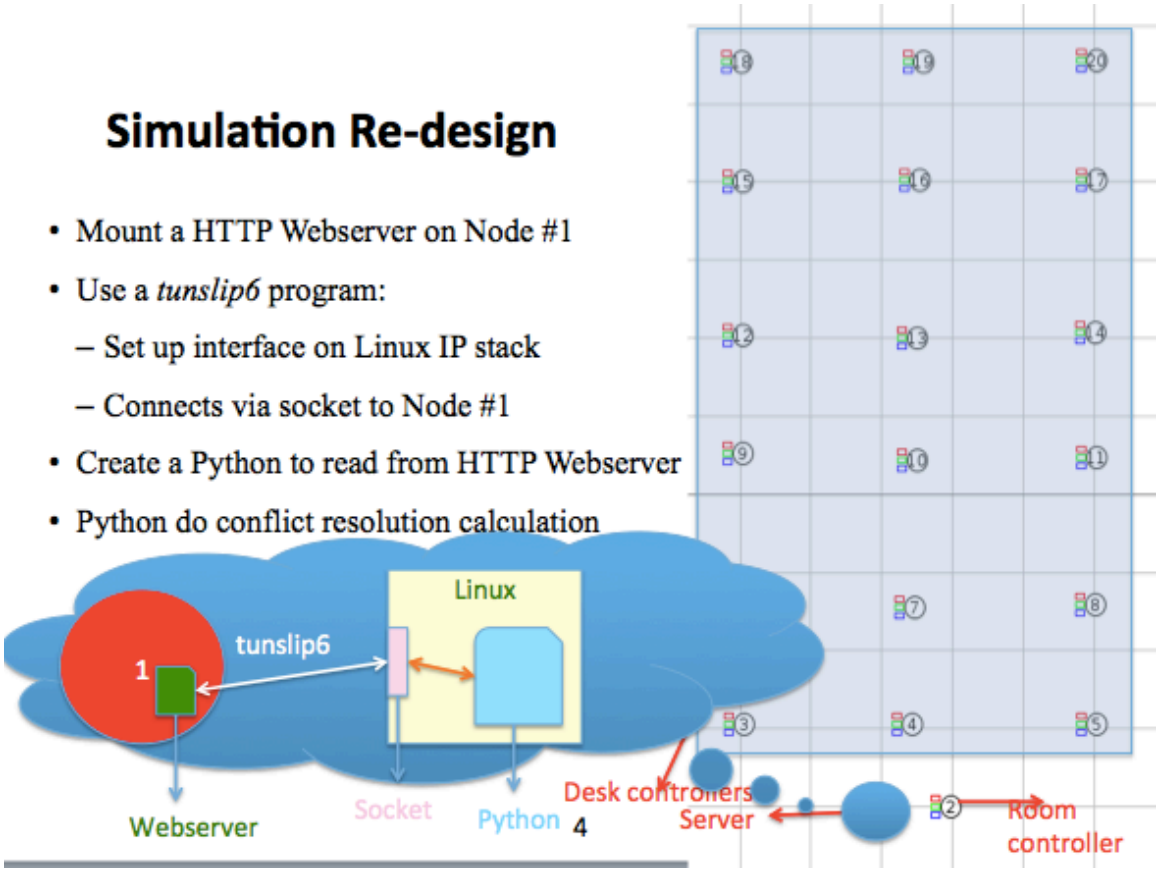

Figure 1. Simulation re-design

By the way, this reference [1] also mentioned how to configure network stack, so I changed RDC layer from ContikiMAC to NullRDC, and changed MAC layer from CSMA to NullMAC, which was also mentioned at the previous meeting with you. However, I am not sure about the motivation to do that, I guess that is because we do not want the node to do radio duty cycling, so the radio will not be powered off. Also, compared with using ContikiMAC+CSMA, NullRDC+NullMAC can reduce the code size of the firmware. But I am not sure my explanation is correct or not, I need your opinion.

Let's go back to the problem itself. Ok, the result of the conflict resolution algorithm can be got, so that I can do a lot of experiments about the algorithm itself, that's good, and that's what I was and am doing. But I figured out that the result cannot be successfully returned back to the node in Cooja. Then I realized that you cannot put the result back to the HTTP webserver, because the code in the Cooja node only post data to webserver, but not receive from the server. Then I tried to use socket in the Cooja node to read the data returned from the Python file. From the test, I can see that the communication between the node and the socket is running well, see Figure 2, the Serial Socket dialog shows that socket->mote: 1219 bytes, mote->socket:1386 bytes, which means it is returning

something back to the node. But when I tried to print out the data that Cooja node received from the Python file, it failed. Here is the current problem, how to see the data that Cooja node receives from the Python file.

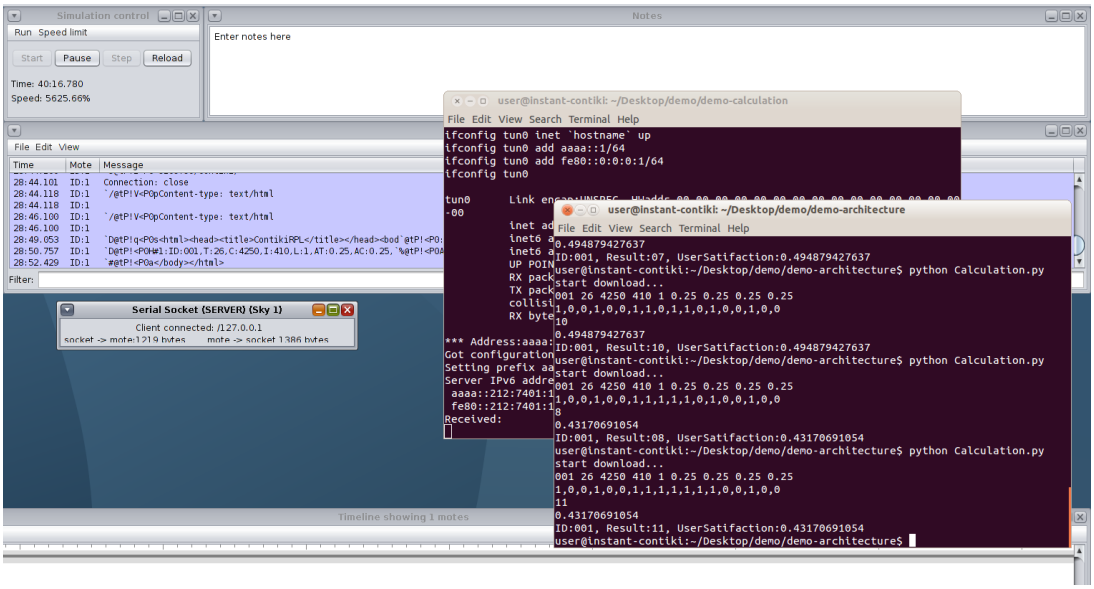

Figure 2. A screenshot of socket communication

I don't know the problem well, but I guess that the problem might be: 1) the "python->socket->node" communication has a conflict with the "node->webserver->socket->python" communication, so actually I tried to use other port instead of the default port, but still not working. 2) something conflict between the socket read function and the print out function in the Cooja node, but it's just an assumption by me because that's very the problem in low level. 3) I think the most probable problem is that the "python->socket->node" communication is not using a correct mechanism, but I cannot come up with other solutions, that's why I need your help.

This is all about the main problem that I had in July, and some other problems/errors have been fixed, or some of them is still being fixed, but not big problem like this. Besides that, I will prepare and comment the code well in order to have a productive meeting on this Friday. Thank you!

Reference

[1] Hands on Contiki OS and Cooja Simulator: Exercises (Part II). https://team.inria.fr/fun/files/2014/04/exercise_partII.pdf