Technical Memorandum (TM)

To: SYSC 3010 Students

From: Dominik Schmidtlein, Student

Subject: Communicating between Android Phone and Raspberry Pi using UDP

Date: October 23, 2015

# Executive Summary

The raspberry pi must be able to send and receive data from an Android phone. The data in question has been limited to strings of maximum size, 1024 bytes. In this solution, the Raspberry Pi runs python, whereas, the Android phone uses Java to send and receive packets. The phone is connected to the router wirelessly. The Raspberry must have a wire connection to the router. The assumption has been made that both devices are on the same local network. The key to the solution is using multithreading, on Android, for sending and receiving functions.

# Method

The method for determining how to send UDP packets between Android and Raspberry Pi was to break the problem down into as small as steps as possible. Pi to Pi connection was already known, however, connection with an Android, in Java, over wifi was all unknown. The first step was to test UDP over WIFI in order to determine which network connections are possible. Only was the problem of Java and Android introduced. This avoided the issue of sending failing, and the cause being attributed to the wrong factor. The two tests are below in figure 1 and 2.

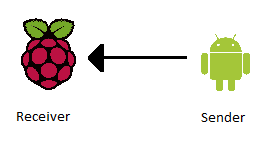


Figure 1: Sending from Android to Raspberry Pi

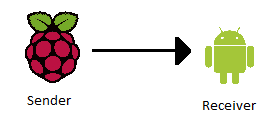


Figure 2: Sending from Raspberry Pi to Android

# Procedure

From the trial project, UDP connections between 2 Raspberry Pis were already successful. The first test was to test UDP connections over WIFI. The computers in CB 5109 tried sending to a wireless laptop, and vice versa, both attempts were unsuccessful. In order to eliminate the complication of sending from the Systems and Computers network, to the general Carleton wifi network, a UDP connection between 2 wireless computers was set up. Both ends were written in Python (see Figure 1 and 2 of Appendix A). The next test was to send from the Android phone to a laptop on the same wifi network as depicted in Figure 1.

After much research and testing it was discovered that the sending function on Android must be executed in a separate thread. The sending function, written in java, can be seen in figure 3 of Appendix A. It is important to note that the “MESSAGE” variable is a String containing the desired message. Also, the “PORT” variable is an int which must have the same value on both the Android phone and Raspberry Pi. The receiving function for the Raspberry Pi, is in figure 4 of Appendix A. The only difference is that the “data” variable did not need to be decoded.

In order to run the sender function on the Android phone, the function needs to be executed in a separate thread. The code is divided into figures 5 and 6 of Appendix A. Figure 5 shows that the sender must implement Runnable in order to be executable in a separate thread. The creation of a separate thread is available in figure 6. In order to test the sending function, the execution was put inside the listener to an onscreen button. The critical line is simply the creation of the thread which is passed the Runnable class. If a second class for implementing Runnable is not desirable, the shorthand in figure 7 could also be used.

Sending from Raspberry Pi to Android is mostly reused knowledge. The operation is shown below in figure 2.

The Raspberry Pi sends, using python, as it did in Pi to Pi communication. There is, however, no need to encode the “MESSAGE” string. Refer back to figure 2 of Appendix A for reference. The Android receiver function is also similar to the sender apart from the call to receive. The function can be seen in figure 8.

# Results and Discussion

The desired connection Android to Pi and Pi to Android were both successful. The key is to execute sending and receiving functions on the Android in separate threads. In retrospect, the code difference between Java and Python is slim. When sending, both languages require initialization of a socket. In python, the initialization of the socket has 2 parameters, whereas in Java the same parameters are passed to the packet. Thus, in python, the explicit creation of a packet is not necessary, in contrast to Java. When sending the packet, both language require a message a port and an IP. However, in Java, the information is inside a packet object whereas in python the packet is created behind the scenes in the send function.

The main difference in receiving between, Java and Python, was that in Java, an empty packet is passed during reception, whereas in Python a tuple was returned.

# Conclusion

The purpose of the research was to send packets between the Android phone and the Raspberry Pi. This was completed successfully. When sending packets using UDP, there must always be a socket, regardless of Java or Python. There must also be packet, although in Python it is created implicitly. In order for UDP to work on Android, the sending and receiving operations must occur in a separate thread.

Appendix A

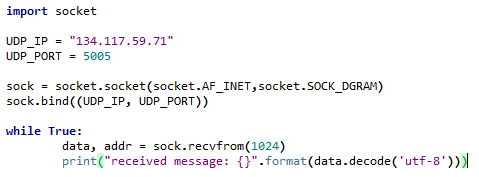


Figure A1: Receiver for Pi to Pi UDP connection

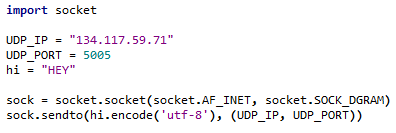


Figure A2: Sender for Pi to Pi UDP connection

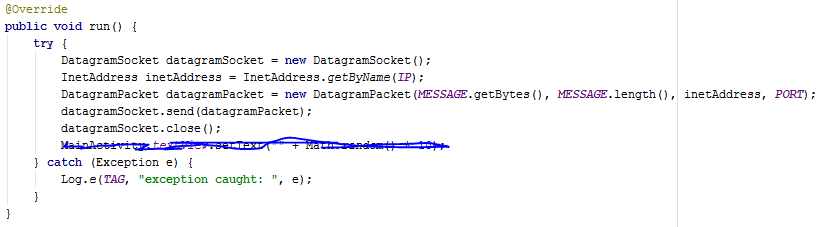


Figure A3: Sending function for Android (Java)

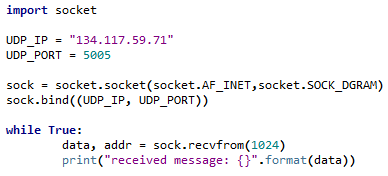


Figure A4: Receiver for Android to Pi connection



Figure A5: Android function must inside Runnable class

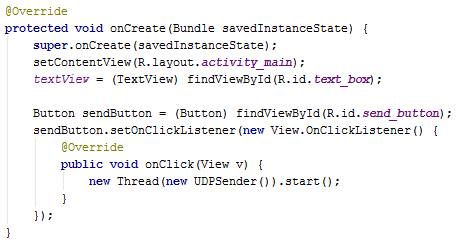


Figure A6: Code for initializing new thread upon send request



Figure A7: Alternative to creating a Runnable class is to put the declaration inside new Thread call

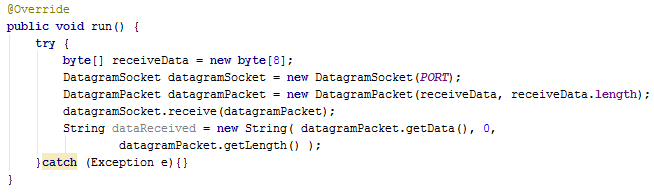


Figure A8: Receiver function for Pi to Android connection