GNU Radio : GUI for Wireless Networks In-the-Loop

Achuth PV (achuthpv@gmail.com, achuthpv@iitb.ac.in)

March 21, 2014

Contents

1 Summary

"Wireless Networks In-the-Loop" (WiNeLo) is a GR-based network emulator which has been under development from 2011 and it was introduced in FOSDEM 2014 [1]. It is a really powerful wireless network emulator which allows users to emulate real time systems in non real time. It substitutes real world RF modules by virtual RF front ends and the real world channel by virtual wireless channel simulator running on a sever along with a control dispatcher, thus providing a emulation environment for the end users to test their software radio applications and all these are virtual components are easily configurable.

The main advantage of the WiNeLo over a conventional network simulator is that there is hardly any effort to switch between the real time and the simulation testing environments due to the WiNeLo sink/source blocks and their GNU Radio companion bindings. This seamless switching would help the developer to shorten development cycle. Along with the normal start and stop simulation feature, it is also providing a capability for pausing the simulation for debugging. This emulator has the capability of supporting dynamic connection and disconnection of clients, which is really a rare feature. With its modular architecture, it supports enhancements quite easily, and it is currently providing support for extending its implementation by providing: basic hardware models, accurate modeling of the timing behavior and a centralized test-management component [7] [9] [8] [10].

Currently the WiNeLo emulator does not have any GUI. So, here I am proposing a project to implement a GUI for the WiNeLo project for the existing framework. Considering the fact that the GR developers are usual with wireless communication and wireless networks, this GUI would surely make their work very easier and intuitive while using the powerful WiNeLo emulator. This GUI which is also going to be build using a modular architecture which can easily be integrated with WiNELo framework, thus providing the capability to easily give GUI support to any enhancement in it. The GUI would help in the visualization of the network structure or layout. Along with providing an ability to look at the node distribution, the GUI would also give the user the ability to configure the nodes, and deploy it randomly and thus change the network structure even during run time. Along with that, it interacts with emulator framework to provide dynamic changes in the channel simulation. This provides user with ability to make sudden changes to emulation depicting dynamic connections (eg: mobile networks) very easily using this GUI. This GUI would also support

integration of GNU Radio blocks to it so that even they can be used for implementing channels, etc, thus making configuration very flexible.
Potential Mentor: Nico Otterbach

2 Tools to be used

GNU Radio [2] is a development toolkit that provides signal processing blocks to implement software radios. It can be used with readily-available low-cost external RF hardware to create software-defined radios, or without hardware in a simulation-like environment.

Wireless Network in-the-Loop [7] [9] [8] [10] is a powerful GR based wireless network emulator. Any software radio application can easily be emulated using seamless switching between the WiNeLo blocks which provides a virtual RF front end to connect application with a virtual channel, and the real world systems.

PyQt/Pyside [3] is Python bindings for the Qt cross-platform GUI/XML/SQL C++ framework.

Scalar Vector Graphics [3] is an XML-based language for describing two-dimensional vector graphics.

QTCreator [3] provides a cross-platform, complete integrated development environment (IDE) for application developers to create applications for multiple desktop and mobile device platforms. With the help of PySide, one can design a GUI in QTCreator, which would generate a XML which can be parsed using Python to configure the WiNeLo components.

Graphviz [4] is an open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. It has important applications in networking, bioinformatics, software engineering, database and web design, machine learning, and in visual interfaces for other technical domains.

3 Benefits

The benefits of having a GUI with WiNeLo is that

- would make emulation using WiNeLo easier and intuitve

- its modular architecture can easily be integrated with WiNeLo framework, thus providing the capability to easily give GUI support to any enhancement in it.
- ability to visualize the network layout and its working
- it makes random node deployment easier
- an existing node can be deleted and a new node can be added easily during run time thus changing network layout during emulation hence supporting the dynamic network connection feature of WiNeLo
- it also interacts with emulator framework to provide dynamic changes in the channel matrix to support the above feature
- each node can be configured easily (providing options to save the configuration at some place (file or DB) and load it)
- easy integration of GR modules to the GUI so that even they can be used for implementing channels, etc, thus making configuration very flexible. These modules can also be used for analyzing purposes (eg. plots).
- tool-tip would help in easy understanding of the current configuration of nodes
- users not comfortable with command prompts can use it easily
- support for drag and drop addition of components
- speedup in testing
- debugging would be much more easier
- logs can be saved easily
- following a modular pattern would help in the GUI to work even if not all the modules completed
- since it is built using Python and C++, it can be easily integrated with GNU Radio

4 Deliverables

The deliverables will be coded in a modular fashion such that it can easily support any enhancement in the future. The figure 1 shows the architecture for the emulator / GUI communication, and the coding will be based on that. The intermediate module will help in easing the communication between the GUI and the emulator providing features like data parsing, etc.

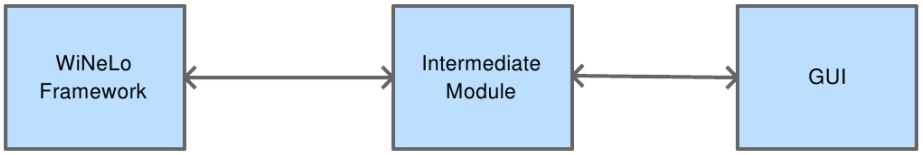

Figure 1: Architecture for GUI/Emulator Communication

First Deliverable: GUI providing ability to visualize: This would provide the user with the visualization of the network layout within WiNeLo. So, the configurations of the nodes and channel matrix, etc would be done within the existing WiNeLo framework only (using the terminal) and the GUI would just capture the details from the simulation server and show it to the user. Here the user will be able to provide the IP of the server to capture the details from the server. The ability to store and show the logs per node would also be provided to the GUI. This GUI would provide visualization of live data from the server.

Second Deliverable: GUI providing ability to edit: Here the user will be able to edit the network layout, configure nodes and the channel matrix from the layout. With the help of Python and GR libraries, the user would also be able to use the GR blocks in the GUI. Each of these components will be there in the side panels from which desired components can be dragged and dropped and then configured. Also the user will be provided with the ability to create new blocks and add it to a project. Here the user will be able to start, stop, and pause the simulation from GUI.

The configurations done by the users will be stored in configuration files (XML), and a call will be made to the simulation framework which would trigger the simulation. Configuration file detail would also be sent to the framework. Parsing of the XML would be done with Python so that each node and the channel matrix are configured as desired by the user within

the simulation framework. The console of each node can also be accessed though this GUI.

Third Deliverable: GUI that supports random node deployment: This is a feature where user will be given ability to add and remove the nodes from the layout during run time. Based upon the positioning of the node on the GUI, the configuration file will be updated. An event will be triggered in the WiNeLo framework to recalculate the channel matrix based upon the modified configuration file during run-time. Thus this supports the feature of dynamic connection of WiNeLo.

Figure 2 would give a rough idea about how the GUI layout will be.

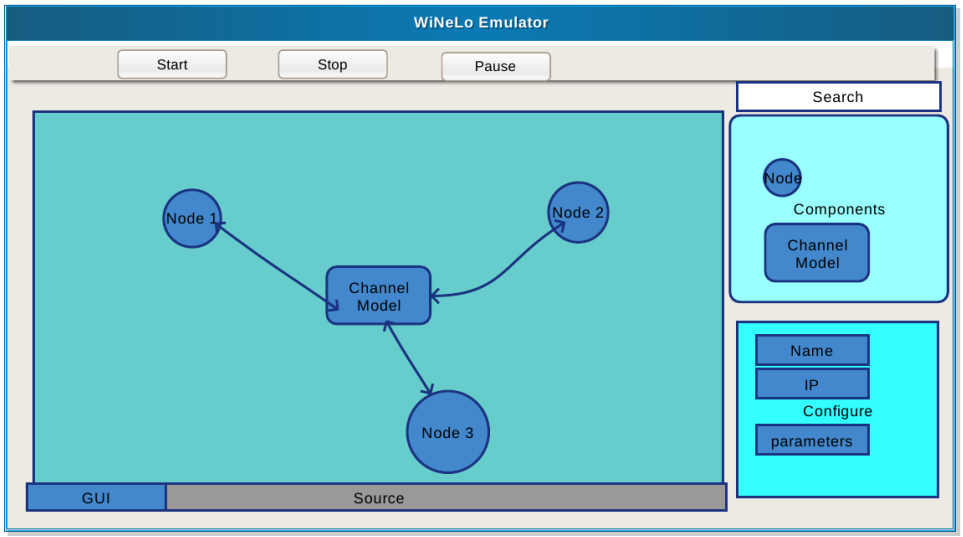

Figure 2: A Rough Layout of GUI

5 Plans

The coding would be done in a test driven manner. I will be using the QTestLib for doing the UI testing. For the GR , Python and C++ codes, I will be following the strict guidelines given in the GNU Radio code practice page. Each line of code will be written with proper naming convention and will be commented properly so that documentation will be easier.

My ideas are inspired from a lot of existing open source network simulators like Omnet++ , GNS3 , etc. So I will be looking up those softwares as the benchmark for the GUI I am planning to build. Since these are open source network simulators, I can even clarify doubts regarding certain implementations by posting queries to their mailing list.

As per the GSOC schedule, the coding period is from May 19 to August 23. I am planning to dedicate around 36 hours per week, and it will be flexible. Following is a rough estimate of the time-line which I would follow. There will be changes in this based upon my discussions with the mentor. Unit tests will be written prior to writing the codes.

- **April 21 - May 5**: Initial discussions with the mentor. Will be taking inputs to make design cleaner. Will be going through the documentations of QT (this includes PySide, SVG), graphviz, and GNU Radio. One of my end semester exam is during the same period.
- **May 6 - May 15**: Will be having a look into the WiNeLo source codes. Will be setting up a environment for testing WiNeLo in my lab and would run few tests to get some practice as well as idea about how it works
- **May 16 - May 18**: Will be checking if any necessary tweaks are to be done in the design of GUI to properly support the WiNeLo framework
- **May 19 - May 25-**: Start coding the first deliverable. This includes creation of the basic components in QT creator. Coding the basic layout and all the interfaces (buttons, etc). Start coding the communication part between the server and the GUI
- **May 26- June 1**: Finish coding the communication part and start working on showing the live data on the GUI
- **June 2 - June 8**: Finish the implementation of first deliverable. Do QA of the first deliverable and fix any issues if present.
- **June 9 - June 15**: Start working on embedding SVG to the QT and thus providing the edit capacity for each node and channel matrix. This also include the provision for dragging and dropping components. After finishing this, start working on the communication from GUI to server which includes starting and stopping the simulation, sending the configuration files to the server and nodes. Here Python will be used

to parse the configuration file from the GUI to configure the nodes and server in the framework

- **June 16 - June 22**: Finish the communication part. Start with the integration of existing GR blocks (analyzing tools, consoles, channel models, etc) to the GUI and also provide ability to write custom blocks
- **June 23 - June 29**: Finish with second deliverable. Prepare for the Midterm evaluation
- **June 30 - July 7**: Do QA of second deliverable.
- **July 7 - July 13**: Fix any issues if present in the second deliverable. Start with the random node deployment feature. This includes adding and deleting the nodes, and changing the layout during run time
- **July 14 - July 20**: Work on recalculating the channel matrix according to the changes in layout
- **July 21 - July 27**: Finish with the third deliverable. Start QA
- **July 28 - Aug 3**: Do QA for the whole product, fix bugs
- **Aug 4 - Aug 10**: Check for any possible improvements. Start code cleaning and documentation
- **Aug 11 - Aug 23**: Finish code cleaning, documentation and do final submission.

My autumn semester would start around July 20th. That is why I have given some buffer time for the development of the third deliverable.

6 Communication

I can contact my mentor, Mr Nico Otterbach via weekly calls and mails. For quick queries, I will contact him via GTalk. I have already subscribed the GNU Radio mailing list, and I am also there in the #gnuradio IRC channel with nick *achuthpv*. I can be even contacted on these email ids: *achuthpv@gmail.com, achuthpv@iitb.ac.in*.

7 Qualification

I am a first year Masters student specializing in Communication and Signal Processing in Department of Electrical engineering, Indian Institute of Technology Bombay, India. My native place is in Calicut, Kerala, India. I am really interested in communication networks, wireless communication and signal processing, also in programming. I have worked a lot on robotics and image processing projects during my undergraduate days and has won a lot of prizes including 1st position in national level. Then I worked in Oracle India Pvt Ltd. for two years as an Applications Engineer (Oracle Fusion group) where I had exposure to various technologies like Oracle DBMS, Oracle ADF, java, ReSTful Services and server framework, JSONs, metadata management, web servers, SVN, ADE (Oracle's version control system) etc. There, I was a good contributor in an internal forum called ADFDi. I did POC for a lot of projects there and also completed projects assigned to me well before deadline. There I had chances to work with teams from various time zones.

I know C\C++ very well, have exposure to Python, CUDA, Matlab and VHDL also git. Although I heard about GNU Radio during my undergraduate days, I had my hands on it for the first time as a part of a course project in Wireless and Mobile Communication taken as a part of my masters. I am also one of the system administrator of the Electrical Department, IIT Bombay. Thus I am comfortable with linux, and also I have access to 30 + machines which I can use for running the simulator and testing my codes. I attend almost every colloquium on communication and signal processing that takes place in the institute. This shows my versatility, my passion for knowledge and since I already have professional experience, I know how to handle each project professionally. I am also a quick learner and also a good team worker.

I am a huge fan of open source communities and always wanted to contribute to it. This was my main motivation to go for GSOC since I want to use this opportunity as the stepping stone into the world of open source coding. I know that it will be a very difficult task. But I have 2 and half month vacation in between which would surely make things much easier and I also have friends who have experience participating in GSOC ready to help. It would be a great feeling for me to see someone using the code which I have developed for GNU Radio.

I also want to continue as an active member of the GNU Radio community even after GSOC and I would like to work more on WiNeLo project.

I also have interests in high performance computation, hardware programming and also image processing.

8 Conclusion

I hope you have got an idea about what I am trying to implement during this summer. It would be great if I can get a chance to work on this project as a part of GSOC. I am open to any questions or suggestions regarding this project.

References

[1] `https://fosdem.org/2014/schedule/event/wireless_networks_in_the_loop/`.

[2] `http://gnuradio.org/`.

[3] `http://qt-project.org/`.

[4] `http://www.graphviz.org/`.

[5] `http://omnetpp.org/`.

[6] `http://www.gns3.net/`.

[7] Gerald Baier and Nico Otterbach. Wireless networks in-the-loop: Speeding up gnu radio development.

[8] Jens Elsner, Martin Braun, Stefan Nagel, Kshama Nagaraj, and Friedrich K Jondral. Wireless networks in-the-loop: software radio as the enabler. In *Software Defined Radio Forum Technical Conference*, 2009.

[9] Sebastian Koslowski, Martin Braun, Jens P Elsner, and Friedrich Jondral. Wireless networks in-the-loop: Emulating an rf front-end in gnu radio.

[10] Nico Otterbach, Martin Braun, and Friedrich K Jondral. Wireless networks in-the-loop: Creating a sdr development environment. *ISWCS 2013*, 2013.