[GSoC'21]Irdest Android Client-Blog IV (A Work Report)

Ayush Shrivastava | Katharina Fey

Saturday 21st August 2021

Prelude

Hi super happy to see you here! It has been an exciting and productive summer from which I learnt a bunch of new stuff and irdest plus GitLab have been gracious on me. The project went through many ups and downs but we made our way fixing the bugs and making things work *as expected*. I hope I'll be able to convey *some* information of the work done by me on a piece of this magnificent software over the course of the summer. Let's begin!

Irdest!?

Okay, so first things first. Let me introduce you with Irdest and what it does to make sure we are on the *same* page and *[irde.st]* doesn't sound completely (we)ird to you, and then we'll progress on the title afterwards. First, I'll try explaining it in a single line,

"Irdest... is a beast!"

Okay no jokes this time(that was no joke btw), going to explanation for real xD, *[irde.st]* is a software suite that allows users to create an internet-independent, decentralized & ad-hoc wireless mesh network. It removes all the dependencies of a user from a specific service and enables users to create a local network mesh of their own. It does not expose data or information of the user. Even the IPs of the peers present in the mesh are not known, they communicate via routers and the entire communication is end-to-end encrypted between the users, thereby increasing privacy in user data. As of now, Irdest supports various functionalities to users like sharing files over the network created, call between users, and messaging.

A Gist

This summer was focused on building the FFI Layer to implement the features supported by the library in the upstream. So if you have been following the initial three posts by me on the same topic, then you must be aware of the fact that the biggest challenge being encountered is compiling library *properly* and linking it to the application in the compile time itself. Apart from this, considerable challenges were about maintaining the robust CI which makes sure we don't break stuff at any point of development process, and the *very* sensitive

FFI layer. We got through these challenges and finally implemented some of the upstream features in the application, but not all. Because with *this* sophisticated setting of the components we need to move forward carefully in order to not break stuff, and with limited time in our hands we decided to implement some of the *very* basic features in the application and write an unbreakable CI for them, from which we can make use of the build artifacts and can keep track where things break.

Work Done

Without going into too much depth of the concepts/thought process and discussion, let's quickly touch upon the work done in the course of this summer of code. You can refer to the previous posts if find yourself interested in detailings of the changes made/steps taken and why and stuff like that.

Compiling the Rust Library

The very first thing I did as a part of the summer of code was fixing the rust library compilation. Initially, the rust library was broken due to the massive refactor and some portion of the huge codebase being left. Due to compilation errors in the rust library(and I being beginner to Rust back then) it took some time to fix the errors, refactor the remaining portion of code accordingly and make it build green. As soon as the rust library was up, the target was to make the application compile and link the library to the application in the compile time. With all these changes being made, a challenge was to write CI for all this cross-compilation setting, which I had never done before.

Writing the CI

Writing the CI for android components including our FFI bridge wasn't *that* tricky, but it did require some good knowledge of cross-compilation, Cargo and obviously android :P. But we ended up implementing that too, and with the *current* state of CI, nothing can break easily and we have awesome and strict checks that compile the components as per the need. We made use of GitLab's one of the greatest and finest works, which is their CI, how they organize and define Pipelines, Jobs, triggering mechanisms and artifacts handling in subsequent and post Jobs. We combined the power of GitLab CI and our own custom docker image *[irdest-android-build-env]* . This made our CI run lightning fast, Jobs that took 11mins to run without any Hi-Fi image being used now finished in 3 to 4 minutes, this was a huge gain and we were able to optimize our CI runs even more via redefining Pipelines, Jobs flow and via introducing the concept of Child Pipelines, another great piece of work by GitLab.

Implementing Features Supported by Library

After all this CI and basic stuff being done, we moved ahead with implementing the functions supported by our rust library in the application. So I implemented the login and registration features, both in the single MR and due to very less

time left in hands, I had to make major UI changes in the same MR, thereby increasing its size, the UI changes were not stellar, but they made the application layouts very responsive and with almost zero dimension hardcodings, everything works like springs, other ones get adjusted automatically, if the change is observed/experienced by any one of all present(for a particular layout).

Some UI Fixes

Also there was a very nasty UI bug that I can remember of, in the Login/Registration screen, in which the screen got split into two components, the login one and the registration one, so in this I setup the optimal fragment transactions and created an abstract layout in the root screen which is empty by default and sets the desired layout file as per the requirements, e.g., it shows the Registration one if clicked on registration button and similar for the others.

Codebase Modernization

In the final days, we moved towards modernizing the application codebase via following some best practices in it and removing the old/deprecated ones :P but sadly this couldn't be merged because of the changes made in the NDK v23 API, which made our cross-compiler plugin incompatible with the project and thereby leading to CI failures, although all of this has now been fixed locally at my own fork, but we wish to implement a stable and *elegant* solution after pondering on the problem for some time. So, along the lines for codebase modernization, the opened MRs included the migration from ol' school Groovy Gradle files for dependency management to human readable Kotlin DSLs, along with some tool version bumps(out of which one was our NDK which I bumped to v23 from v21 xD, yeah I can see 'ya a bit sad, it hurts ;( ) and some changes in Kotlin scripts we were able to compile the library directly from the Android Studio itself, which previously was a great PITA and we had to *manually* compile the library. The next MR targeted the migration from legacy view scans to ViewBinding, increasing the application performance!

Ah, I am not going to list *all* the MRs opened by me in the summer here, but if interested you can give 'em a look here:
[we/irdest/merge_requests?author=s-ayush2903]

Futher Possible Improvements

Well there are really a *bunch* of improvements that can be made in the existing codebase! Let me help you think of few:

- Writing Unit tests for the features implemented by far
- Writing Instrumented tests for UI flow implemented by far
- Making the application support many/some more functions that the library supports
- Running instrumentation tests on CI
- ~~Fixing the NDK v23 incompatibility with our cross-compiler plugin~~

The last entry was a joke, ignore it xD

Acknowledgements :)

Well we finally arrive here. A huge thanks to my amazing mentor, Spacekookie <3, who was always there to help me out when stuck and shared their valued thoughts on what directions we need to take for the project. Discussions with them have always been super super insightful and let me ponder for a while about their thought process in figuring out the solutions. A big thanks to you again! Nextly, this project would never have been possible without the organization Freifunk where I got accepted as a GSoC'21 student to work on one of their project. It was a truly amazing experience where I learnt a lot of *new* stuff and met people having same interests, which made the project and discussions more involved, productive and helpful. Thanks to all. Although I'm a *bit* disappointed about the very limited time we had to work on the project and couldn't make it to the level we thought at a point of time. But anyways, super happy after working on Irdest!

If you're interested in the project, you may find following useful, where I discussed about the tech involved, in detail:

- #1 Overview blog – *[@LaTeX]* | *[@freifunk-blog]*
- #2 Phase I report – *[@LaTeX]* | *[@freifunk-blog]*
- #3 Phase II report – *[@LaTeX]* | *[@freifunk-blog]*

Cheers, until next time we meet!
∼Ayush Shrivastava /