GSoC ‘17 Project Proposal

Organization: OpenWISP

OpenWISP 2 Network Topology

Basic Information

Name and contact information:

- **Name:** Rohith A. S. R. K.
- **Email:** rohith.asrk@gmail.com
- **IRC Nickname:** rohithasrk
- **Phone number:** +91 9100149957
- **Github:** rohithasrk
- **Skype:** live:5dae8a13c338f257
- **Video Chat:** https://appear.in/rohithasrk

University and Current Enrollment:

- **University:** Indian Institute of Technology, Roorkee
- **Field of Study:** Electronics and Communication Engineering (Batch of 2019)

Meeting with mentors:

- Reachable anytime easily through email or IRC. A well planned session if required:
- Mondays, Wednesday, Fridays between 2:00 pm - 12:00 am IST and any day from 6:00 pm - 12:00 am IST.

Coding Skills

I am very comfortable with all the technologies that are to be used in this project. I can write (in order of proficiency):

- Python: Programming in python, Django web framework.
- JavaScript: JQuery, AngularJS, Vanilla JS.
- Java/ C++ (Can revise if needed).

Development Environment

- Ubuntu 14.04 LTS, Python 2.7 installed.
- Highly proficient in using Vim, Emacs and Sublime text.

- Good familiarity with virtualenv/ vagrant.

Version Control

- Very strong concepts of Git. Used Github and Gitlab a lot.
- Also comfortable with Mercurial and SVN.

About Me:

I am a 19 year old, sophomore currently enrolled in Electronics and Communication engineering at IIT Roorkee. I have developed a passion for programming and web development in my freshmen year and from then, most of my time goes into reading and writing software. I have been contributing to open source regularly since about six months now - looking over to repositories of the products I use/ come across, trying to give my part of contribution back to the organization whose product has been an asset to me.

I have the experience of working closely with a team as I am an active member of **Information management Group** at IIT Roorkee, a bunch of passionate enthusiasts who manage the **institute main website**, internet and intranet activities of the university.

The biggest project that I've worked on is LecTut. LecTut is an intranet study portal of IIT Roorkee where professors and students can upload and share the course material with everyone. My work involved managing the platform and working on the backend infrastructure that runs all the services. This included developing APIs using Django and Django REST Framework, implementing elasticsearch using haystack and coding the frontend in AngularJS. A lot of challenges were solved in implementing this. My role was to understand the needs of the system, design the infrastructure from scratch, implement it and make a tool to automate the batch addition process every year.

This is the first time I might contribute to a big open source project.

Pre-GSoC Involvements

Here is a list of issues/ PRs I've worked on so far in **OpenWISP** or **related** projects:

- **#32 (merged):** Made the hint text bolder
- **#35** (**merged**): Fixed a grammar error in models.py
- **#101(mergeable):** Improved string representation of sortedm2m relationships
- **#100(mergeable):** Added swp files to gitignore
- **#102:** Opened up an issue and suggested a way to fix PEP8 style errors

- **#36:** Opened up an issue to suggest setup virtual environments using Vagrant.
- **#18:** Fix all style errors using automated tools

Other than OpenWISP, I've contributed in other organisations too. These are the list of contributions in them.

Zulip: An open source chat application (Django, JavaScript)
- **#3153 (merged):** This PR involved migrating the use to send_mail function from EmailMultiAlternatives
- **#3416 (merged):** This PR involved setting up better file naming conventions in Zulip and make necessary changes.

InfosecIITR: The Information security organisation of the university
- **#3 & #8 (both merged):** Added security tools for web pentesting.

Previous experiences with OpenWRT:

We are given LAN cables in each and every room of hostel in IIT Roorkee. Accessing the internet requires **802.1X** authentication from the client. I tried setting up a router in my room and I couldn't access the internet because the software in my router didn't support **802.1X auth.** I've then set up OpenWRT in my router and it helped me solve the problem. This got me interested and I started exploring the field of networking.
> Do you have a router at home on which you can flash OpenWRT in order to test OpenWISP?
Yes, I have a router which is OpenWRT compatible on which I plan to do my testing.

Motivation

My motivation for GSoC this year is getting started with the open source community. GSoC is a great program for introducing organizations with prospective contributors and when I saw this project, I thought this is something that aligns with my area of interest.
Networking has always been my field of interest and I'm naturally inclined towards contributing to the project. I started to interact with the community and got more and more support from the prospective mentors. And that is when I gained confidence about the project. The thought of me being able to contribute to something big and in turn learn a lot motivates me to work with OpenWISP for the SoC. I strongly believe that with my skills and the mentorship, I'll be able to contribute well to the organisation.

Project Details

OpenWISP 2 Network Topology

Introduction

Being able to collect and visualize the mesh network topology is the best way and yet is crucial for learning about a particular network. This is very beneficial to someone working with and managing hundreds of routers/ access points.

Features

- This app is will basically be **wrapper** to django-netjsongraph as openwisp-controller is to django-netjsonconfig.
- A web interface with multi-tenancy features and extending django-net
- A mesh network visualizer in the admin panel for users of particular organization to view the network topology.
- A basic monitoring system where a user can go back in time and visualize the status of the network.
- Signals whenever a link in the topology is changed.

Possible mentors - Claudio Pisa, Valerio Coltrè, Federico Capoano

Measurable Outcomes

The main measurable outcomes of the project are as follows:

- Implement a reusable django app named **openwisp-network-topology** This app would depend on **django-netjsongraph** and shall avoid duplication of code.
- Shall improve abstraction of **django-netjsongraph** and extend it in **openwisp-network-topology.**
- Add multi-tenancy features by using **openwisp-users**
- Create a signal at instances whenever the status of the link changes
- Implement a way to save daily snapshots of the network topology
- Add a switcher in the visualizer that allows to go back in time and see one of the previous daily snapshots
- Add a way to customize the theme of the visualizer
- Add a **netdiff** parser for OpenVPN that works on both of its status formats
- Improve **django-netjsongraph**, **netdiff** and **netjsongraph.js** when necessary to implement features upstream.

- Document all the features in the README
- Prepare a setup.py script that will be used to install the python package of the reusable django app
- Achieve a test coverage more than 80%
- Provide documentation using python-sphinx

Achieving Project Goals

The entire project can be broadly classified into four phases through completion:

Phase 1: Deliverable before the first Mid-Term evaluation.

- **1.1 Add multi-tenancy features by using openwisp-users to make it usable in openwisp2.**
- **1.2 Visualize network topology in the admin.**
- **1.3 Prepare a setup.py script that will be used to install the python package of the reusable django app.**
- **1.4 Writing tests for all the functions written till now. I plan to learn about test driven development and follow.**
- **1.5 Improve abstraction of django-netjsongraph by adding base classes and mixins.**

Phase 2: Deliverable before the second mid-term evaluation.

- **2.1 Implement a way to store snapshots daily of the network topology.**
- **2.2 Switcher in visualizer to go back in time and see the snapshots.**
- **2.3 Add a way to customize the theme of the visualizer and frontend work.**

Phase 3: Deliverable before the final evaluation.

- **3.1 Add a netdiff parser for OpenVPN that works on both of its status formats**
- **3.2 Provide documentation using python-sphinx**

Wishlist: Can be implemented if all the above tasks are completed successfully and the programme is yet to end, also desired to be taken up post GSoC.

- **4.1 Improve UI/ UX of the app**
- **4.2 Improve test coverage upto maximum.**
- **4.3 Add additional features in django-netjsongraph, netdiff or netjsongraph.js.**

Phase 1

1.1 Add multi-tenancy features by using openwisp-users.

Most of the work has already been implemented in the django-netjsongraph package. The models can be classified into three main categories named **Node, Link and Topology**. Link is a communication channel between two nodes, a source and a target. And a topology can be thought as a **'mesh island',** which has all details on which we base our analysis.

- Now, this feature in the app aims to provide authentication and relate a topology to an organisation. An organisation can have multiple topologies, but a topology can be related to a single organisation.
- An organisation, as implemented in **openwisp-users**, can have many users in it.
- Users can view topologies of only their organisation, whereas a super admin can view and edit all the topology info.
- To implement, we define a model named **Topology** which inherits from **BaseTopology** that was defined in **django-netjsongraph** and has an attribute named **organization** which is a foreign key to the **Organization** model. A very basic implementation can be shown like this.

```
from django_netjsongraph.models.topology import BaseTopology
from openwisp_users.models import Organization

Class Topology(BaseTopology):
      organization = models.ForeignKey(Organization)
```

- Therefore, there shall be a **login** mechanism where users login and it shows the panel where various topologies of a particular organisation are listed. We shall **filter topologies** based on **organisation** of the user authenticated and provide them with various features from then.

1.2 Prepare a setup.py script that will be used to install the python package of the reusable django app.

- This would ease the setup work for both user and a developer.
- There are some specific set of rules that are to be followed in order to accomplish this and is given in the django docs which I tend to follow.
- Can be found **here.**

1.3 Write texts to maximise the test coverage.

- Writing tests is an essential part of building an application. It not only allows easy debugging while development, but also in production or when any new feature is introduced.
- Will be using the django.test module, i.e., the unittest framework of python and write tests, following proper guidelines. I think the best way of writing tests is before writing a particular view/ function and will implement this way.

1.4 Visualize network topology in admin panel

- As openwisp2 doesn't have a frontend yet, we need to devise a way to show the network topology visualization in the admin panel.
- This can be accomplished by extending the admin and defining a class named **BaseTopologyAdmin** and adding views which can render the templates.
- This will appear like a pop up window and show the network topology visualization.
- This can be done by making an API call and rendering response as it happens in **django-netjsongraph.**

1.5 Improve abstraction of django-netjsongraph by adding base classes and mixins and organising files

- We aim to make django-netjsongraph's classes extendable just like it happens in django-netjsonconfig.
- The code needs to be organised and categorised in files.
- Adding abstract and base classes for each common class shall be done as implemented in django-netjsonconfig **here.**
- More features shall be simultaneously added as per requirements.

Phase 2

2.1 Implement a way to store snapshots regularly and also whenever the status of the link changes

- One of the desirable features is to keep track of network changes and also being able to re visit a particular day to view the snapshots of the network topology.
- This feature can be divided into two parts,
- Regularly saving the snapshots at a particular time of a day.
- Saving the snapshot whenever link status changes.
- First of all, we define a **save_snapshot()** function in the views mapped with a url named **save_snapshot** which would do the task of saving a snapshot.
- In order to save the snapshot, we could,

- Save the data which is obtained in NetJSON format as a string in the database.
- For this, there'd be a model that we'd define, by the name, say **network_topology_snapshot** which would have an attribute **organization** foreign key to the **Organization** model of **openwisp_users**, **data** being the string of the data in NetJSON format and **datetime_created** field which can get automatically appended.
- In order to save snapshots regularly, a cronjob shall run in openwisp2 which shall define the frequency of snapshots.
- Whenever the link status changes, we need to send a signal to take a snapshot of the earlier network topology. This could be achieved by defining **a new link** model which inherits the **BaseLink** model defined in **django_netjsongraph,** and overrides the save function of the BaseLink model to call a **notify()** function.
- We shall implement a configurable way of taking snapshots with this signal and the default settings shall be decided later.

2.2 Switcher in the visualizer to go back in time and view snapshots.

- As we've saved snapshots at various points and also regularly, one of the expected features would be to revisit back and check them out.
- We can do this by just adding two buttons, forward and backward which would help traverse days. There would be a view by name **topology_by_date(date)** with date in arguments which would get all the network data of a particular date from the database.
- This data is now sent to the visualizer to display.
- This is the simplest implementation that I've mentioned. We should definitely think of improving the UI/ UX of the user.

2.3 Add a way to customize the theme of the visualizer and frontend work.

- There shall be a settings button on the page and a drop down where a user can customize the theme of the visualizer by
- Changing the colors of background using a **color picking tool**.
- For this effect to be permanent, we can store the user preferences in the database and make an **AJAX** request every time on load.
- A model named **user_color_preferences** or so can be defined in order to store the hex values/ color codes in the database. This model will have a **user** parameter which is a foreign key to **User** model defined in **openwisp_users**, and a **color** parameter to store the color code of user's preference and automatically a **datetime_created** field added.

Phase 3

3.1 Add a netdiff parser for OpenVPN that works on both of its status formats.

- There needs to be a file named openvpn.py in the parsers directory of netdiff repository. Add these formats in the topology options in **django-netjsongraph.**
- This aims to extract details from the output of different formats of OpenVPN.
- An example output can be shown as follows

```
OpenVPN CLIENT LIST
Updated,Mon Mar 27 18:47:04 2017
Common Name,Real Address,Bytes Received,Bytes Sent,Connected Since
claud43nodo1,80.181.191.1:50268,53875636,192011412,Sun Mar 26 09:06:20 2017
LeMonacelle,2.226.154.66:44846,5164751,19264301,Mon Mar 27 16:10:59 2017
Syskrack,79.26.49.2:60616,56928467,189355476,Sun Mar 26 09:06:19 2017
Kali2,2.226.154.66:55438,52681920,193489019,Sun Mar 26 09:06:20 2017
mirko,79.36.196.24:65039,1057697,245684199,Sun Mar 26 09:06:20 2017
buda,79.12.108.6:62026,1061815,245676448,Sun Mar 26 09:06:23 2017
pomezia,95.251.243.132:50275,1058494,245684089,Sun Mar 26 09:06:21 2017
ROUTING TABLE
Virtual Address,Common Name,Real Address,Last Ref
aa:f7:ef:8f:55:52,Syskrack,79.26.49.2:60616,Mon Mar 27 18:45:29 2017
8a:0b:ac:35:42:c6,LeMonacelle,2.226.154.66:44846,Mon Mar 27 16:11:01 2017
a6:26:32:97:8b:6c,claud43nodo1,80.181.191.1:50268,Sun Mar 26 23:14:10 2017
c4:6e:1f:ff:02:23,Kali2,2.226.154.66:55438,Mon Mar 27 18:45:29 2017
GLOBAL STATS
Max bcast/mcast queue length,11
END
```

The first part contains all clients connected. The second part contains the details of all the links, source being the virtual address and the target being the vpn server. We need to convert this into a NetJSON NetworkGraph format, show that we can visualize it. This should be simple as we just need to parse the next. One simple implementation has been given in this link.

3.2 Provide documentation using python-sphinx

- Sphinx is a great tool and renders beautiful documentation with a variety of output formats.
- I believe in documentation of the code simultaneously after writing the code. Never had much experience with python-sphinx though. Can easily go through it when required.

Proposed Timeline

I have my end semester examinations between 23 April - 1 May. I will be inactive between 10 April to 2 May for exam preparations, which is way off the timeline.

I believe I have enough fuel to get started on my goals - as a result to my involvement with the organization for almost two months now. So I will be setting grounds early to avoid stopgaps during the actual GSoC period. There are a couple of weeks before the actual timeline where I would make sure that I have done all the homework. All dates mentioned here are weeks - starting from Monday and ending on Sunday.

3 April - 9 April (Homework week)

- Research more about django-netjsongraph, netdiff and netjsongraph.js
- Learn writing efficient tests.
- Work on unmerged pull requests.

10 April - 7 May (Inactive period)

- End semester exams will be near so, I'll be bit inactive, though always available on email or any other means of communication. At times, I'll try solving an issue or play around with OpenWISP 2 and networks for recreation.
- Between 2 May - 8 May, I'll be packing up from the University campus and moving home, so mostly inactive these days.

8 May - 14 May (Community bonding - I)

- Interact with the members of the community and discuss about the project with the mentor.
- This would be the time I'd utilize properly and learn the concepts which I ain't so familiar with so that I don't lag once the coding period begins.
- Start fixing issues in django-netjsongraph to improvise the code and get experience with the code design.
- One of the major changes in django-netjsongraph would be making more abstract classes, as implemented in django-netjsonconfig.

15 May - 21 May (Community bonding - II)

- Discuss about the workflow with the mentor and make mock ups/ wireframes of the design going to be implemented.
- Decide goals and non-goals of the project and get a clear idea of the features to implement.
- Fix issues in the repositories and add new features to get a quick head start in developing the application.

22 May - 28 May (Community bonding III)

- Finalizing all the technology stack going to be used and backend design.
- Learn more about making reusable apps and mastering them by making a few test applications.
- Set up initial directory structure to get a kick start to development from the coming week.

29 May - 4 June (Week 1)

- Start work with all enthusiasm. Week 1 is dedicated to setting up basic repository and adding multi-tenancy features to the application.
- This involves adding a setup.py file that will be used to install the python package of the reusable django app.
- This would involve coding the backend of the feature. Basically all the API work and also writing tests simultaneously.

5 June - 11 June (Week 2)

- Develop visualization of network topology in admin panel. The details of which have been described above.
- Document all the features written till now in a README file.

12 June - 18 June (Week 3)

- Testing our admin panel visualization feature and multi-tenancy and make sure it is fully functional.
- Write tests for the features developed in the previous week.
- Obtain a good test coverage.

19 June - 25 June (Week 4)

- Add a basic models and views to store snapshots i.e., complete the backend of the feature.
- This is a buffer week for preparing the code to be presentable for mid-term evaluation.
- My target in this week will be to put a firm pencils down on work done so far, updating documentation, writing tests and code cleanup.

Milestone reached: Phase 1 completed.

26 June - 2 July (Week 5)

- Write tests for **save_snapshot** feature. Make sure the function pass all the tests.

- Signal when a status of a link changes and about the snapshot being taken.
- Code APIs for a **switcher** feature and add tests for the same.

3 July - 9 July (Week 6)

- Make the switcher fully functional, i.e., adding necessary html and css and integrating with the APIs made.
- Document the newly added feature and check the test results.
- Add backend of the feature to customize the theme of the visualizer.

10 July - 16 July (Week 7)

- Add the color picker tool or the other ways of user being able to customize the theme.
- Integrate with it's backend.
- Write tests for the same and improve the coverage.

Milestone reached: Phase 2 completed

My institute re-opens on 18 July, till now I was totally free throughout the day so could easily code for 8 - 10 hours a day. I have kept enough buffers to clear backlogs if any and I am confident that I can work upon the timeline I stated so far. I can work for about 7-8 hours a day from now.

17 July - 23 July (Week 8)

- Make **netdiff parsers** for OpenVPN and make a PR to **netdiff.**
- Add version1 and version2 parsers as options in the Topology model in **django-netjsongraph** and make a PR.
- Write tests for the same and document features accordingly.

24 July - 30 July (Week 9)

- Refining the work done so far on parsers.
- It is not necessary that everything goes as planned out and there might be unavoidable delays due to a nasty bug hidden from eyesight.
- Finalize the content of deliverables after Mid-Term Evaluation and update documentation.
- Document the code using python-sphinx and render documentation according to the organization's preferences.

31 July - 5 Aug (Week 10)

- Document using python-sphinx if something was missed.
- Debugging and improving the test coverage will be the top priority.
- Improving UI/UX being the next.

Milestone reached: Phase 3 completed

6 Aug - 12 Aug (Week 11)

- **Tentative wrap up - cosmetic refinements.**
- Update documentation - write unit tests wherever missed out and do a general code cleanup. Make sure there is nothing left undone and everything is tidy.

13 Aug - 19 Aug (Week 12)

- Prepare content for the end term evaluation, including deliverables from the wish list.

20 Aug - 29 Aug (Conclusion)

- Buffer week and conclusion of GSoC.

Availability

My vacations start from 3 May and end on 15 July. The official GSoC period is from 5 May to 21 August. I can easily devote 40-50 hours a week till my college reopens and 30-40 hours per week after that. I'm free on weekends and mostly free on Wednesdays. I intend to complete most of the work before my college reopens.

Other than this project, I have no commitments/ vacations planned for the summer. Also, I don't plan on doing any internships this summer. I shall keep my status posted to all the community members on a weekly basis and maintain transparency in the project.

After GSoC

> Are you interested in working with OpenWISP after the GSOC ends?

Yes. I'm interested as I always was. I've been contributing to OpenWISP for over a month now and I've really loved the experience. I got familiar with the community and I believe I've learnt a lot interacting with the prospective mentors. I feel this kind of mentorship is necessary. I'll be an active member in the community and keep contributing. My motivation would always be that I'd be able to contribute to something big and widely in use application. This gives me a lot of satisfaction.

> If we get new business opportunities to build new features, would you be interested in occasional freelance paid work?

Well, I'm always ready for some work that'd teach me something new and cool. I'd be really happy to associate with the community for any business opportunities. I shall take time out of my tough schedule at the college and willingly work for the upliftment of the community.

References

1. **OpenWISP GSoC '17 Ideas page**
2. **Documentation of django-netjsongraph**
3. **Network Topology Visualizer: Django-Netjsongraph**
4. **Documentation of unittest framework**
5. **Mesh networking**
6. **Github repositories of django-netjsongraph, netjsongraph.js, netdiff, django-netjsonconfig, openwisp-controller, openwisp2.**

****