GSOC ‘18 PROJECT PROPOSAL

ORGANIZATION: OpenWISP

OpenWISP 2 IPAM

BASIC INFORMATION

Name and contact information:

- **Name:** Rohith A. S. R. K.
- **Email:** rohith.asrk@gmail.com
- **IRC/Gitter/Github:** rohithasrk
- **Phone number:** +91 9100149957
- **Country/ Region:** Roorkee, India

University and Current Enrollment:

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

ABOUT ME

I am a third year undergraduate student studying Electronics and Communication Engineering at IIT Roorkee. I’m passionate about web, networking and software development. I got introduced to software development in my freshman year and since then I’ve been trying out new technologies and contributing a variety of projects. Most of my time goes into reading and writing software. I am an avid internet user and a regular reader of the tech forums like HackerNews, Reddit etc.

I’ve successfully completed Google Summer of Code 2017 with OpenWISP and have been an active part of the community since then. I maintain django-netjsongraph, openwisp-utils, openwisp-network-topology, and netdiff. I’ve been a very active Google Code In 2017 - 18 mentor for OpenWISP. I have all the resource available which are essential for the project. My motivation for GSoC this time is working on challenging projects that would enhance my skills and also be of great use to the network community. I strongly believe in this project that I have chosen to create huge impact. Apart from that, I’d also be helping the other GSoC students which would improve the grounds for being a mentor in the coming years.

PRE-GSOC INVOLVEMENTS

Here is a list of PRs/ discussions made so far after the GSoC application period has begun.
In django-freeradius:

- **#110 (open)** - A working implementation of enforcing session/ bandwidth limits on users.
- **#109 (open)** - Batch add users feature.
- **#108 (issue)** - Add more attributes to the RadCheck model to impose more restrictions.
- **#106 (open)** - Altered migration files to add support to freeradius3.
- **#105 (merged)** - Improved docs for configuring freeradius dev environment.
- **#104 (open)** - Removed repetition of id declaration.
- **#97 (issue)** - Import shared code from openwisp-utils
- **#96 (issue)** - Add openwisp-admin theme to tests/ django-freeradius
- **#94 (issue)** - Add support to django > 2.0 to requirements-test.txt

Other major contributions to OpenWISP can be checked out at GSoC 2017 with OpenWISP. Statistically, they could be summarized as

- 43 commits (4288++, 3039- -) in netjson/django-netjsongraph
- 44 commits (2633++, 1217- -) in openwisp/openwisp-network-topology
- 8 commits (1396++, 32- -) in openwisp/openwisp-utils
- 10 commits (230++, 35- -) in ninuxorg/netdiff
- 3 commits (114++, 5- -) in openwisp/ansible-openwisp2
- 1 commit (49++, 1- -) in gregmuellegger/django-sortedm2m
- 1 commit (15++, 0- -) in openwisp/openwisp-users
- 2 commits (6++, 1- -) in openwisp/django-netjsonconfig

PROJECT DETAILS

Introduction

IP address management modules are vital when it comes to managing networks of big organizations. In these scenarios, being able to visualize and monitor the subnets comes handy. This module shall be a wrapper of a new reusable package named django-ipam, which shall be initialized during the program.

Features

- IPv4 and IPv6 address management
- Section/ Subnet management with nested subnets
- Automatic free space display for all subnets
- Visual display for a particular subnet
- IP request module
- RESTful API for CRUD operations
- Search for an IP or a subnet
- CSV import/ export of subnets and their IPs

Possible mentors - Federico Capoano, Marco Giuntini

Measurable Outcomes

- Create a reusable package named django-ipam with all the above features
- The package shall have all its components abstract to support further extensibility.
- Build a module openwisp-ipam which shall wrap django-ipam with multi-tenancy.
- Provide documentation for django-ipam and openwisp-ipam using sphinx.
- Achieve a test coverage more than 90%.
- Add openwisp-ipam as an optional module in ansible-openwisp2.

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 Initialize django-ipam and prepare a setup.py script that will be used to install the python package of the reusable django app.**
- **1.2 Abstract models and model tests along with registration on admin site.**
- **1.3 Automatic free space display for all subnets.**
- **1.4 RESTFul API for CRUD operations.**
- **1.5 Add IP request module.**

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

- **2.1 Visual display for a particular subnet.**
- **2.2 Search for an IP or subnet.**
- **2.3 CSV import/ export of subnets and their IPs**
- **2.4 Release django-ipam 0.1 and fix issues upstream.**
- **2.5 Initialize openwisp-ipam and prepare a setup.py script that will be used to install the python package of the reusable django app.**

Phase 3: Deliverable before the final evaluation.

- **3.1 Pull in django-ipam into openwisp-ipam and add multi tenancy on top of it.**
- **3.2 Release openwisp-ipam 0.1 and fix issues upstream.**
- **3.3 Integrate openwisp-ipam in ansible-openwisp2 as an optional module.**
- **3.4 Provide documentation using python-sphinx.**
- **3.5 Release ansible-openwisp2 and fix issues if any.**

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.

- **Add additional features in django-ipam and pull them in openwisp-ipam. Depending on the feature-requests and suggestions of the openwisp2 users.**

DETAILED EXPLANATION

Phase 1

1.1 Initialize django-ipam and prepare a setup.py script that will be used to install the python package of the reusable django app.

- This would involve creating a repository named django-ipam and preparing all the necessary scripts needed for a PyPi package.
- This would be similar to the work done in the initial-commit of the network topology module (openwisp-network-topology).

1.2 Abstract models and model tests along with registration on admin site.

- This task includes finalizing the initial database structure of the application.
- The very basic models for the application would be **IPAddress, Subnet** and **Section**.
- These models shall inherit themselves from the respective abstract class named AbstractIPAddress, AbstractSubnet and AbstractSection.
- Basic model structure can be checked out at rohithasrk/ipam/models.py

1.3 Automatic free space display for all subnets.

- A property named subnet_usage shall be defined in the Subnet model which calculates the free space available and returns it out.
- It can be checked out at rohithasrk/ipam/models.py#L33

1.4 RESTFul API for CRUD operations.

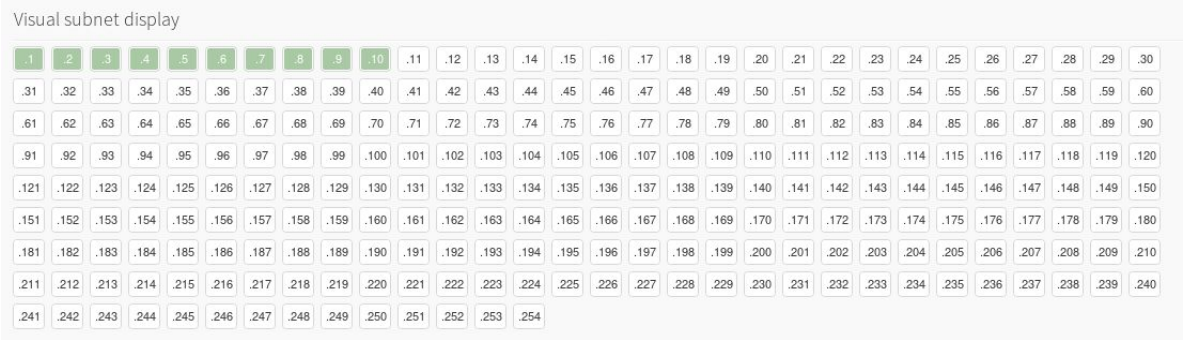
- For Create, Retrieve, Update and Delete we shall use the default django-rest-framework APIs.
- This would be something similar to that implemented in rohithasrk/pastebin/views.py
- Would have session authentication and permission settings of rest_framework on the APIs of Update and Delete.

1.5 Add IP request module.

- A widget named 'IP Request' shall be added as an optional widget which will have two fields Subnet and IP which shall be used to make a request.
- For implementing this, a model by the name `IPRequest` shall be added which shall have fields like `ip_address, hostname, description, requested_by, comment, admin_comment, and accepted`.
- This module shall have its respective CRUD API.
- Only the admin can accept IP requests. This can be added using the default permission classes of the rest framework.

Phase 2

2.1 Visual display for a particular subnet



- This would involve showing the IPs that are used in a different shade and a auto generated boxes depending on the amount of free space.
- Javascript would be handy in this scenario.
- Each element can be stored in a div generated accordingly depending on the array index.
- Tutorials for implementing this can be found here**.**
- This is similar to displaying the page for seat booking in movie booking sites.

2.2 Search for an IP or subnet.

- A search API which uses queries of the default ORM provided by django.
- Filtering mechanisms which can be checked out from here can also be implemented.
- This could be ajaxified as per user's requests.

2.3 CSV import/ export of subnets and their IPs.

- This task is similar to the one implemented in django-freeradius#109.
- Can be implemented the same way.

2.4 Release django-ipam 0.1 and fix issues upstream.

- Wrap up the initial version and release it fixing any issues found on testing.

2.5 Initialize openwisp-ipam and prepare a setup.py script that will be used to install the python package of the reusable django app

- Same as the task 1.1 but this time for a different package.

Phase 3

3.1 Pull in django-ipam and add multi tenancy on top of it.

- Multi tenancy is having a single instance which runs of server and serves multiple tenants. openwisp-users is a package which is used to implement multi-tenancy in all the openwisp-modules.
- The AUTH_USER_MODEL would now be openwisp_users.User and an organization field would be added to each and every model of openwisp-ipam while pulling in django-ipam.

- There exists a OrgMixin in openwisp_users which upon inheritance adds up a `organization` field to all the models.
- For implementing multi tenancy in the admin interface, the reusable mixins from openwisp-utils shall be used. These are **MultitenantAdminMixin, MultitenantAdminMixin, MultitenantRelatedOrgFilter**.

3.2 Release openwisp-ipam 0.1 and fix issues upstream.

- Wrap up the initial version and release it fixing any issues found on testing.

3.3 Integrate openwisp-ipam in ansible-openwisp2 as an optional module.

- It may need the implementation to be done in a sub role that becomes a dependency of ansible-openwisp2.
- This would be similar to the way openwisp-network-topology has been added to openwisp2 as an optional module.

3.2 Provide documentation using python-sphinx

- Sphinx is a great tool and renders beautiful documentation with a variety of output formats.

3.3 Release ansible-openwisp2 and fix issues if any.

- Make a public announcement about the new addition of the openwisp-ipam module.
- Fix issues that arise in the production and ensure the project works well.

PROPOSED TIMELINE

2 Apr - 8 Apr	Explore https://phpipam.net/ and improve the drafted database structure.
8 Apr - 15 Apr	
16 Apr - May 6	Inactivity due to university examinations and academic work.
May 7 - May 13	Initialize django-ipam, prepare a setup script and finish of with the models of the application.
May 14 - May 20	
May 21 - May 27	Work on free space of a subnet and develop REST APIs for CRUD operations.
May 28 - Jun 3	IP Request module and extensive unit tests for the same.
Jun 4 - Jun 10	
Jun 11 - Jun 17	Visual display of a subnet.
Phase 1 evaluation	
Jun 18 - Jun 24	Visual display of a subnet. Write frontend tests for the same too.
Jun 25 - Jul 1	Search an IP or subnet. CSV import of IPs and Subnets.

Jul 2 - Jul 8	Release django-ipam, Init openwisp-ipam and integrate multi tenancy on top of django-ipam.
Jul 9 - Jul 15	
Phase 2 evaluation	
Jul 16 - Jul 22	Release openwisp-ipam. Integrate openwisp-ipam as an optional module with rest of openwisp modules.
Jul 23 - Jul 29	
Jul 30 - Aug 5	Release ansible-openwisp2 and fix issues/ add features requested by the users. Conclude GSoC by making final additions/ fixes.
Aug 6 - Aug 12	
Phase 3 evaluation - Conclusion	

Availability

My vacations start from 7 May and end on 15 July. The official GSoC period is from 14 May to 14 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. 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. It has almost been an year I've been contributing to OpenWISP. It has been a great ride. I've learnt a lot of skills over this time and believe that I can now independently put it to the real world use. I aim to research more about wireless networking and feel like OpenWISP has a lot to give. Apart from that, the best part of working with OpenWISP is the goals of the projects and the mentorship of it's developers. With the community growing continuously, I feel responsible for all the projects I'm a part of. I've been one of the most active members on the chatroom and have been contributing in almost all possible ways.
> If we get new business opportunities to build new features, would you be interested in occasional freelance paid work?
Yes, if I feel it would be beneficial to both the community and myself. There is always learning involved in working on projects with deadlines.

REFERENCES

1. OpenWISP GSoC '18 Ideas page
2. https://phpipam.net/

****