Google Summer of Code 2021

Revamp Netengine and add its SNMP capability to OpenWISP Monitoring

---

Personal Information

Identification -

- **Name -** Saurav Shrivastav
- **Nationality -** Indian
- **Location -** Pune, Maharashtra, India
- **Timezone -** IST (UTC +5:30)

Contact -

- **Email** - sauravsrivastav103@gmail.com
- **Phone** - [redacted]
- **Github/Gitter.im username** - Saurav-Shrivastav

Educational Information -

- **Institute -** Thapar Institute of Engineering and Technology
- **Degree -** Bachelor of Engineering
- **Major -** Computer Engineering
- **Current year -** Sophomore (2nd Year)

Working hours -

- Reachable on Gitter, email, or contact number in UTC 0330 - 2030 hrs
- Working hours -
    I. UTC 0430 - 0730 hrs (IST 1000 - 1300 hrs)
    II. UTC 0930 - 1230 hrs (IST 1500 - 1800 hrs)
    III. UTC 1530 - 2030 hrs (IST 2100 - 0200 hrs)

About Me

Educational -

I am a second-year undergraduate student pursuing my Bachelor's degree in Computer Engineering at TIET, Punjab, India. I was introduced to the ways of programming in my high school when I started learning Java. Since then, I've been trying out various technologies by taking up new projects, participating in hackathons, and contributing to Open Source.

My complete Academic Resume can be viewed here.

Volunteer Experience -

- Google Developer Student Club - Core Member
  - It is a group of passionate developers and designers. This is where I was introduced to the world of Open Source.
  - At DSC, I got a chance to develop software for thousands of students and configure our servers and deploy our apps into production.
  - Learned how to solve issues efficiently and also how to collaborate better with my team.
- Microsoft Learn Student Chapter - Core Member
  - Learned a lot of time management skills and the ability to deliver, punctually, under pressure.

Freelance Experience -

- CYO gyms
  - Implemented an API with Django-graphene.
  - Created a subscription-based Django channels application.
- AnalyticWare
  - Implemented Google-OAuth and integrated a payment gateway for the project.

Open Source Experience -

- BotsFramework -
  - Implemented continuous delivery using TravisCI to build the plugin upon a new release.

    - Scraped websites like Medium.com, Dev.to, etc. with BeautifulSoup, and built a minimal flask server that provides a JSON response of a list of articles for the particular tag/keyword to be searched.
    - Integrated the scraper flask service to build slack and discord bots that send daily articles onto the channels.
    - Worked as the project maintainer and managed various issues and pull requests for the project.
- DSC Official Website
    - Migrated the project to pipenv.
    - Worked on some feature endpoints and models.
    - Documentation updates

Experience with OpenWRT -

I got to know about OpenWRT when I started researching OpenWISP for GSoC'21 in early March. I have OpenWRT installed in a VM, that I use for learning and testing purposes.

I have a router at home that can be used for flashing OpenWRT.

Motivation -

Networking is one of the fields that has always intrigued me, and I am interested in exploring it. OpenWISP is the perfect platform for me to delve into networking and know my way around it. I have been working with Django and Python for a considerable amount of time. It acted as one of the reasons that motivated me to select OpenWISP as my GSoC organization.

Open-source Contributions | OpenWISP -

- Pull requests Authored

PR	Contribution	Status
#233	Fixed - Exception raised on opening non-existing user ID	**Merged**
#234	[change] Make email lowercase when adding a new user	**Merged**
#404	[docs] Updated "Installing for development" instructions in the README	**Merged**

#412	[fix] VpnClients destroyed on any change in configuration or templates	**Merged**

- Issues involved with

PR	Contribution	Authored	Status
#228	Solved the issue in #233.	No	**Closed**
#227	Solved the issue in #234.	No	**Closed**
#399	Made a PR to solve this issue in #412.	No	**Closed**

Currently working on Rest endpoints for Openwisp-monitoring in issue #290.

GSoC Project

Revamp Netengine and add its SNMP capability to OpenWISP Monitoring

The objective of this project is to add support for SNMP (Simple Network Monitoring Protocol) to OpenWISP Monitoring by using Netengine, a python library that aims to make it easy to access monitoring information via different protocols.

The project is divided into 2 major phases -

Phase 1 -

- Revamp Netengine
- Update tests

Phase 2 -

- Update Documentation
- Add Netengine's SNMP capability to Openwisp-monitoring

Possible Mentors -

Federico Capoano, Gagan Deep

Measurable Outcomes -

- Revamp Netengine:
  - Port the code to python >= 3.7

    - Revamp the backends for OpenWRT and Ubiquiti to make them compliant with NetJSON DeviceMonitoring Specification.

- Update tests in Netengine:
    - Update the unit tests to reflect the changes and to ensure that all tests pass.
    - Change the tests to use mocks.
    - Ensure test coverage stays above 95%.
    - Add openwisp-controller like QA-checks.
    - Create a test build on Github actions.

- Update Documentation
    - To reflect the changes that are introduced in the project.
    - Write a new readme.
    - Also, provide basic developer documentation.

- Add Netengine's SNMP capability to Openwisp-monitoring
    - Modify OpenWISP Controller to allow setting the management IP from the web UI
    - Add an SNMP check-in OpenWISP Monitoring that pulls the monitoring information and creates the device status and charts

Project Details -

Phase 1 -

- **Port code to python >= 3.7**
    - Setup a tool like coverage.py and ensure good test coverage.
    - Update the code
        - Run python3-futurize over the test suite first and visually inspect the diff to make sure the transformation is accurate.
        - After I have transformed the test suite and verified that all the tests still pass as expected, then the application code will be transformed knowing that any tests which fail are a translation failure.

        - Since tools cannot automate everything, other things will be transformed manually.
    - Check for dependencies that might block the transition with caniusepython3.
    - Create a runtest.py script to run the test suite.
    - Create a test build on Github actions and check that everything works properly.

- Revamp the **backends for OpenWRT and Ubiquiti** to make them compliant with **NetJSON DeviceMonitoring Specification.**
    - Will have to modify the code for the backends to make them compliant with the format mentioned here.
    - A similar implementation is present in netengine-utils, it can be used as a reference.
      Snippet from `if-config` parser:

```
def to_netjson(self, python=False, **kwargs):
        """ convert to netjson format """
        result = []
        for i in self.interfaces:
            netjson = OrderedDict((
                ('name', i['name']),
                ('mac', i['hardware_address']),
                ('mtu', i['mtu']),
                ('ip', [])
            ))
            if not netjson['mac']:
                del netjson['mac']
            # add ipv4
            if i['inet'] and i['mask']:
                netjson['ip'].append({
                    # creates an ipv4_interface object correctly
                    'address':
str(ipaddress.ip_interface(u'%(inet)s/%(mask)s' % i))
                })
            # add ipv6
            for ip in [i['inet6'], i['inet6_local']]:
                if ip:
```

```
                netjson['ip'].append({
                    'address': ip
                })
        result.append(netjson)
    # can return both python and json
    if python:
        return result
    else:
        return json.dumps(result, **kwargs)
```

- **Modify tests** to incorporate changes made in the previous step.
  - The current tests do not check for NetJSON compliance, I will have to modify them and make sure they pass for correct output.
  - I found tests that perform a similar function and will be a perfect reference to achieving this goal - L257-L362 in tests/utils/ifconfig.py.

- Change the tests to **use mocks**.
  - unittest.mock will be used to mock the behavior of physical devices. I will have the mock return any output that the device would return, or even raise an Exception when called.
  - Mock instances will be created using utilities like the Mock class and patch decorator, return values will be set for the mock instances and the outputs will be tested. The patch() decorator handles patching module and class level attributes within the scope of a test, it'll be used along with "sentinel" for creating unique objects

- Setup **QA Checks**
  - This will be a relatively easy task since it has been already implemented in other OpenWISP modules and it is only a matter of replicating similar tests for netengine.
  - The next step would be to add support for the QA checks in the Github actions build.

Phase 2 -

- **Clean up -**
  - This step would involve cleaning up the repository and removing unnecessary code.

- **Update Documentation -**
  - Update docs to reflect the changes introduced in this project.
  - provide good documentation for netengine in form of a README.rst (using ResStructuredText), explaining how to install, run tests, configuration details, screenshots, and explanation of the main features. Proper markdown rules will be followed.
  - provide basic developer documentation, explaining how to install and run tests

- Modify OpenWISP Controller to **allow setting the management IP from the web UI**
  - The management_ip field is now read-only, to support active SNMP checks we have to allow users to edit this field.
  - If a device has SNMP credentials (present in the openwisp-controller/Credentials model), an edit field will be displayed in the panel, this can be achieved by modifying the DeviceAdmin class.

- Add an **SNMP check in OpenWISP Monitoring** that pulls the monitoring information and creates the device status and charts
  - SNMP Credentials will be created, they will then be used to authenticate connections with a new connector class in the openwisp-controller/connections module.
  - An SNMP-check class will be coded to implement active SNMP checks in openwisp-monitoring.
  - Add relevant tests to ensure that the checks work properly.

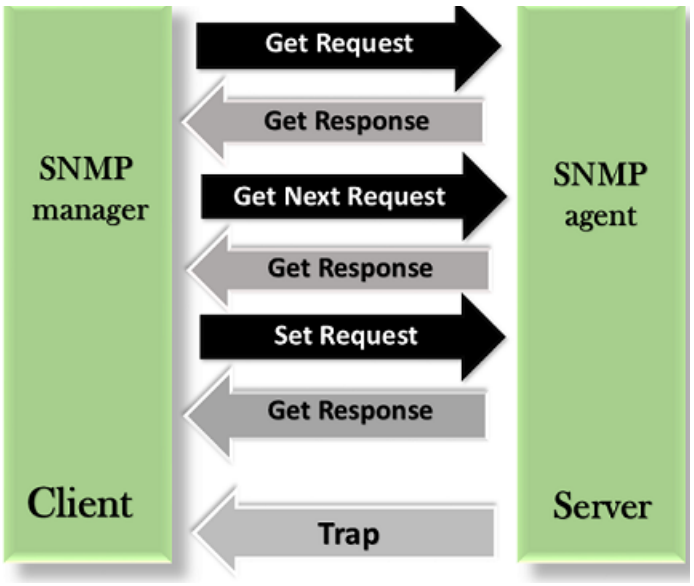

Fig 1.1 - The SNMP model

Proposed Timeline -

April 14th - May 16 After submitting the GSOC application.	• Continue regular contributions to the repositories - find solutions to issues, make pull requests, review other PRs. • I will complete my work on issues like #290 • Learn more about OpenWISP and networking. • Will begin minor contributions to Netengine.
May 17 - June 7 Community Bonding Period	• Interact with the mentors and set up feedback loops. • Continue to refine the plans for the project in consultation with the mentors, I have mentioned the path and way for the solutions in the proposal but they are flexible and can change as per the common view with the mentor. • Get involved with other members of the community. • Extensive research on implementations and methodologies for the set goals to make the solutions easier

	I will **begin working** in this period. ● Spending 8-10 days on setting everything up and porting the Netengine library to python >= 3.7. ● Start revamping the backend for OpenWRT

PHASE - 1

Week 1 & Week 2 - June 7 - June 19	Complete revamping the backend for OpenWRT
Week 3 > June 21 - June 26	Revamp the backend for Ubiquiti to make it compliant with NetJSON DeviceMonitoring Specification
Week 4 > June 28 - July 3	● Complete revamping Ubiquiti backend ● Modify tests to incorporate changes made in the previous step. (most of them will be completed simultaneously, this step ensures that all tests are in place)
Week 5 > July 5 - July 10	● Change the tests to use mocks. ● Setup QA checks.

PHASE - 2

Week 6 > July 12 - July 17	Clean up and update documentation.
Week 7 > July 19 - July 24	Modify OpenWISP Controller to allow setting the management IP from the web UI
Week 8 > July 26 - July 31	Add an SNMP check in OpenWISP Monitoring that pulls the monitoring information and creates the device status and charts
Week 9 & 10 - Aug 2 - Aug 16	Buffer Period

Apart from this, I will be maintaining a **2-part blog** (for each phase), writing about my work and the entire workflow.

I have also kept some buffer days in my timeline, giving me time to complete any delayed tasks.

Availability -

During the GSoC period, I can spend around **40-50 hours** per week on the project. I don't have any pre-planned vacations or plans during the break and will be available full time.

My college will reopen somewhere around mid-August, and I can devote around **30-35 hours** per week once that happens. I intend to complete most of the work before my college reopens and use the buffer days in case of any backlogs.

I shall keep my status posted to all the community members every week and maintain transparency in the project.

After GSoC

I have learned a lot and picked up a majority of my skills by contributing to the OpenWISP project over the spring season and even after the Google Summer of Code, I plan on continuing my contributions to this organization, by adding to my past projects and working on open issues.

The highlight of my experience with OpenWISP has been the fast-planned development and mentors' active involvement. With the community growing continuously, I feel responsible for the project of which I am a part. Having picked up a lot of developing skills, my major focus would be to develop mentorship skills so that I can give back to this community by helping other people navigate around and reviewing their contributions.

- If we get new business opportunities to build new features, would you be interested in occasional freelance paid work?
    - Yes, I am interested in doing occasional freelance work. In the past, I have worked with a few startups for freelance paid work.
- Will you help to maintain your implementation for a while?
    - Yes. For me, the open-source contribution is not just restricted to Google Summer of Code. I would love to keep working on the developments and enhancements of the project.

Why OpenWISP?

As I mentioned previously, I am quite passionate about Networking and the tech surrounding it, which turned out to be one of the major reasons for me choosing OpenWISP as my mentor organization. When I began contributing to minor issues for OpenWISP, I genuinely liked the coding standards and the workflow followed by the organization. Moreover, the community is very active and I hope to learn a lot from the mentors and other peers in the community, helping me enhance my networking knowledge, building quality products, and improving my coding skills on the way. I appreciate OpenWISP's commitment to the community and the work it has been doing and I surely want to be a part of its journey.

Contributing to OpenWISP will help me to equip essential real-world software skills as well as soft skills which I believe will help me a lot in the longer run.