NANYANG TECHNOLOGICAL UNIVERSITY

FINAL YEAR PROJECT
FINAL REPORT

SCE15-0672
PerfSONAR Data Visualization

Name	Mohamed Nazri Bin Mohamed Yahya
Matriculation Number	U1320756D
Supervisor	A/P Lee Bu Sung
Examiner	A/P Zheng Jianmin

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

NANYANG TECHNOLOGICAL UNIVERSITY

SCE15-0672

PerfSONAR Data Visualization

Submitted in Partial Fulfilment of the Requirements
for the Degree of Bachelor of Computer Science
of the Nanyang Technological University
by

Mohamed Nazri Bin Mohamed Yahya
U1320756D

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

2016

Abstract

Networks are an integral part of society. Other than being the backbone of the internet, it is also an important tool to share resources amongst educational institutions. As networks continue to grow, it becomes unpredictable and difficult to maintain or troubleshoot in an event of failures. With that, network measurement tools are becoming increasingly important. One of such tools is PerfSONAR, a tool developed by Internet2, ESnet, Indiana University and GEANT through an international collaboration.

PerfSONAR is an open source network measurement tool having the capability to measure a wide variety of network metrics. Such tools empower network engineers as it provides valuable insights on the networks, and to assist in managing end user expectations. With the tool becoming dominant in the networking space, there is a lack of visualization to display the data in an informative manner.

PerfSONAR Data Visualization aims to create a web based application for the network engineers to observe the network from an overview perspective. Current visualizations built for PerfSONAR are only able to display from the perspective of a single instance of PerfSONAR. The web based application also provides the feature to highlight networks with abnormal network measurements which are considered as being above the threshold defined by the Upper Fence.

Beyond the features, PefSONAR Data Visualization looks at adopting proper frameworks for both Front-End and Back-End development to support code maintainability and extensibility for future enhancements.

Acknowledgements

I would like to express my appreciation and gratitude to Associate Professor Lee Bu Sung, for providing guidance, sharing valuable insights, and patience throughout the entire period of this project.

I would also like to express appreciation and gratitude to Mr Simon Peter Green from SingAREN on providing the virtual machines, and his expertise in networking.

Table of Contents

Chapter 1 Introduction

1.1 Background

In today's context, the internet plays a tightly integrated role, becoming society's indispensable tool. It is a gateway to many of our important services such as online banking, access to government services and many more. The internet comprises of interconnected networks where millions of computers are connected. This allows any computer on the network to communicate with other computers. As such, networks are important as they form a communication pipeline between these computers [**1**].

Networks are also an important collaboration tool. It brings educational institutions together to share knowledge and to explore new frontiers. An example would be the Large Hadron Collider (LHC) Computing Grid. The LHC is the world's largest and most powerful particle collider built by the European Organization for Nuclear Research (CERN) from 1998 to 2008 [**2**]. LHC aims run experiments where particles collide at high speed to test theories of particle physics and high-energy physics. The outcome of these tests would generate a massive amount of data, reaching approximately 50 petabytes in a year. The staggering amount of data prompted CERN to establish the LHC Computing Grid, an international collaborative project that will store, distribute and analyze the data. The LHC Computing Grid is based on Grid Computing where a collection of computers from multiple locations are linked in a network thus allowing the computers to work in tandem to reach a common goal. The LHC Computing Grid span across 174 facilities in over 40 countries. As such, the network supporting the LHC Computer Grid plays an important role in transferring the data [**3**].

With networks playing a pivotal role not only for CERN, it is crucial that networks are in good standing to serve its needs. However, due to unforeseen circumstances, failures may happen in a network. There are predominantly two types of failures, hard failure and soft failure. A hard failure in a network is easily identifiable with the obvious signs of the network being down. On the other hand, soft failures are harder to discern. Networks with soft failure will continue to run however not up to the capabilities it was designed for. This requires additional tools to capture network metrics and to troubleshoot in order to determine a soft failure [**4**].

PerfSONAR (**Perf**ormance **S**ervice **O**riented **N**etworking Monitoring **Ar**chitecture) is an open source network measurement toolkit. It contains multiple tools such as traceroute and ping to collect a wide variety of network metrics [**5**]. Having a variety of datasets on hand allows network engineers to observe and troubleshoot networks in an event of failures. However, the current tools that are used to show these data are one dimensional, showing only the metrics between the two end points of the network at a time. This does not allow the network engineers to observe the network on a grand scheme of things. Having the ability to observe multiple networks or an overview of networks allows the network engineers to distinguish patterns of abnormality in the network and assist in troubleshooting the fault.

1.2 Objectives

The objective of this project is to develop a web based application that will utilize the data and metrics collected by PerfSONAR to create visualizations for the network engineers. The visualization aims to display an overview of all the nodes measured in PerfSONAR and to highlight networks with readings above the threshold defined by the Upper Fence. Upper Fence is defined as upper limit of the data given by the equation of,
Upper Fence = Third Quartile + 1.5 * (Interquartile Range).

1.3 Scope

The scope of the project is as follows,

- The project utilizes only the Traceroute and Latency (Round Trip Time) data to create the visualizations.
- The project will only retrieve the network measurement data over the last seven days.
- The project will flag network measurement readings if it exceeds the threshold which is defined by the Upper Fence.
- The project is a web based application to be run in a modern browser with JavaScript support.
- The project utilizes a back-end server that will provide additional support for the web based application.
- The project will pull data from a centralized Measurement Archive situated in Japan.
- The project will display the following types of visualizations
    a. Traceroute visualization highlighting Round Trip Time values above threshold
    b. Traceroute visualization highlighting changes in path
    c. Latency (Round Trip Time) visualization

1.4 Project Requirements

This section of the report covers the applications and libraries that were used to build the visualization. The project consisted of the Front-End development or also known as Client-Side development where users are able to view and interact with the visualization directly, and Back-End development or also known as Server-Side development where additional logic resides to support the Front-End.

1.4.1 Front-End Development

Name	Version	Description
WebStorm	2016.2	A JavaScript Integrated Development Environment (IDE) to support the Front-End development by providing features such as code assistance, debugging and integration with Version Control Software such as Git.
AngularJS	1.4.9	A JavaScript Front-End web application framework by Google.
Angular Chart	1.0.0	A charting library based on ChartJS repackaged for AngularJS.
ChartJS	2.0.1	A JavaScript based charting library. This is a dependency for Angular Chart.
Angular Animate	1.4.9	An animation library for AngularJS.
Angular BootStrap	2.0.1	A front-end framework to maintain the structure and design for the website repackaged for AngularJS.
Angular Cache	4.6.0	A cache library to manage caches in browsers. This is wrapper for the HTML5 Web Storage API.
Angular Loading Bar	0.9.0	A library that adds interactive loading bars onto the website.
Angular Mock	1.4.12	A library that provides support to AngularJS to conduct unit testing.
Angular Toastr	2.0.0	A library that adds toast notification to the web application.
Bootstrap	3.3.6	A Front-End framework that defines the structure and design of the website. This is a dependency for Angular Bootstrap.
Font Awesome	4.6.3	A library consists of fonts and icons.
Cytoscape	2.6.3	A library to create and visualize network graphs.
jQuery	2.1.4	A JavaScript library to allow easy manipulation of Document Object Model objects in HTML pages.
MathJS	3.5.3	A JavaScript mathematical library containing common mathematical functions.

1.4.2 Back-End Development

Name	Version	Description
PyCharm	2016.2	A Python Integrated Development Environment (IDE) to support the Back-End development by providing features such as code assistance, debugging and integration with Version Control Software such as Git.
Python	2.7.11	A programming language used to support the Front-End.
Flask Web Framework	0.11.1	A Python based micro web framework to handle requests from the Front-End.
DNSPython	1.14.0	A Python based DNS library that is able to conduct reverse DNS lookup based on IP addresses.

1.5 Overview of Report Organization

1.5.1 Chapter 1: Introduction

This section covers the background, objective and the scope of the project. It allows the reader to understand the motivation of the project along with the expectations.

1.5.2 Chapter 2: PerfSONAR

This section of the report provides an overview of PerfSONAR. Having an understanding of PerfSONAR is crucial as it plays a fundamental role in this project.

1.5.3 Chapter 3: Review of Current Visualizations

This section of the report looks at the eight golden rules in designing an interface and reviews existing visualization created for PerfSONAR.

1.5.4 Chapter 4: Front-End Development

This section explains the tools and libraries used in developing the front-end aspect of the web based application and demonstrate the importance of the tools.

1.5.5 Chapter 5: Back-End Development

This section explains the tool and libraries used in developing the back-end aspect of the web based application. This section will also demonstrate how the tools were used and how the back-end supports the front-end of the web based application.

1.5.6 Chapter 6: Deployment

This section covers the deployment of the web based application. It covers the installation procedure and steps to deploy the project on a Linux environment.

1.5.7 Chapter 7: Conclusion

This section concludes the project.

Chapter 2 PerfSONAR

The principal idea of the project is to develop a web based application that is able to create visualizations from the network metrics measured and collected by PerfSONAR. As such, PerfSONAR is a vital component to the project and having an understanding of PerfSONAR and its functionalities is essential. Other than understanding PerfSONAR, it is also important to understand the network metrics as this help to steer the direction of the project.

PerfSONAR (**Perf**ormance **S**ervice **O**riented **N**etworking Monitoring **Ar**chitecture) is an open source network measurement toolkit develop through an international collaboration with Internet2, ESnet, Indiana University and GEANT. It contains a comprehensive set of network measurement tools allowing it to measure a wide variety of network metrics. These network measurement tools can be scheduled to run regularly with the results are stored in a time series database embedded within PerfSONAR. With this, PerfSONAR aims to provide insights of the network to network engineers assisting to manage end to end usage expectations.

2.1 Components of PerfSONAR

As a comprehensive network measurement toolkit, it contains various components under its belt. The components are implemented based on a Service Oriented Architecture, as suggested by the name [**6**]. A Service Oriented Architecture is a software design architectural pattern that promotes loose coupling between the various components. A loosely coupled design pattern consists of well-defined entities with proper interfaces to allow interaction with each other. This design pattern promotes maintainability as each of the entities does not have dependencies and extensibility as new entities can be easily added as part future enhancement. With regards to PerfSONAR, each of these entities are known as a service. Having this understanding of the structure of PerfSONAR allows to tap on the necessary services to build the web based application.

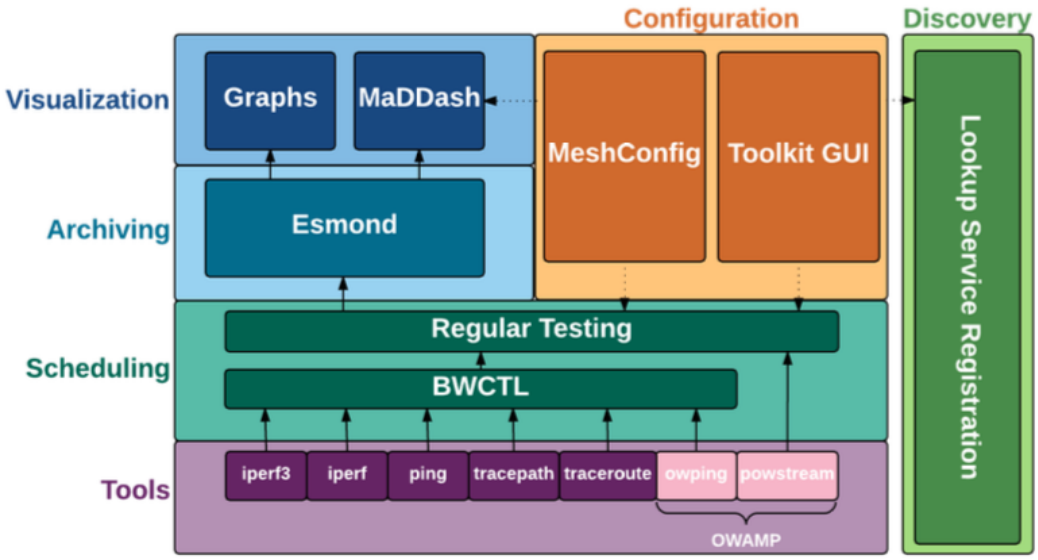


Figure 1. Components in PerfSONAR. [**7**]

The figure above shows the breakdown of components in PerfSONAR. In building the web based application, the relevant components to understand falls in the section of “Archiving” and “Tools”. Esmond, also commonly referred to as the Measurement Archive [**8**], is a system

developed by ESnet to store time series data collected by the network measurement tools in PerfSONAR. Esmond also provides a Representation State Transfer (REST) interface, allowing external applications such as the web based application in this project to retrieved the stored data. REST is a form of architectural style that allows computer systems on the internet to exchange information in an interoperable manner. The details of using the REST interface will be discussed in the implementation section of the report.

The “Tools” component of PerfSONAR are responsible for conducting the network measurement tests. As shown in Figure 1, PerfSONAR contains multiple tools such as iperf3, iperf, ping, tracepath, traceroute, owping and OWAMP. As the web based application will only create visualizations from the traceroute and latency data, the tools applicable are traceroute, tracepath and ping. Interpreting these data is important as it steers the design choices and considerations in building the visualization. This is further discussed in the following sections.

2.1.1 Traceroute and Tracepath

Traceroute and tracepath are commands part of the Linux operating system. These commands have been embedded as part of the tools in PerfSONAR. Both of these commands are similar, in the sense it is able to trace a path to the desired destination. However, the tracepath command may not be available in all Linux distribution. In an event that tracepath is not available, PerfSONAR will fall back to the traceroute command [**9**].

Determining the path to the destination is able to reveal information of the intermediary nodes. Having this information assist network engineers to determine if the packets are taking the optimal path, fully utilizing the capabilities of the network. The path to the destination can also assist network engineers to troubleshoot issues such as loss of packets or high latency, possibly due to the failures in intermediary nodes, leading to a change of path [**10**].

In traceroute, packets are sent to the destination in order to determine the path. With each packet sent, a Time-To-Live (TTL) value is set, decremented with each intermediary router it reaches. Once the TTL value reaches zero, an Internet Control Message Protocol (ICMP) message of “time exceeded” is sent back to the source [**11**].

The first packet is sent with TTL value set to one. Upon reaching the first router, the value of TTL is decremented, making the value of TTL zero thus an ICMP message of “time exceeded” is sent back. With this, the router’s address is noted along with the time taken, determining the round trip time. After ICMP message arrived, traceroute will increment the TTL value to two and repeat this procedure. This is repeated until an ICMP message of “Port Unreachable” is received [**11**]. This determines that the packet has reached the destination host or the packet has reached the maximum number of hops.

nazri @traceroute Analysis: ~$ traceroute perfsonar.net
traceroute to perfsonar.net (207.75.164.248), 30 hops max, 60 byte packets

1	203.30.39.254 (203.30.39.254)	2.568 ms	0.473 ms	0.202 ms
2	et-4-3-0.537.rtswlosa.net.internet2.edu (198.71.45.170)	183.070 ms	182.916 ms	182.680 ms
3	et-1-0-0.111.rtr.hous.net.internet2.edu (198.71.45.20)	215.231 ms	215.087 ms	214.759 MS
4	et-7-3-0.4070.rtsw.kans.net.internet2.edu (198.71.45.17)	230.052 ms	229.942 ms	229.673 ms
5	et-5-3-0.4070.rtsw.chic.net.internet2.edu (198.71.45.14)	240.349 ms	240.872 ms	240.703 ms
6	ae4x2058.chcg-lvl3-600w.mich.net (192.122.183.45)	240.713 ms	240.814 ms	240.439 ms
7	irbx69.anar-cor-cath.mich.net (198.108.22.98)	250.314 ms	250.507 ms	250.548 ms
8	ae0x56.anar-cor-mitc.mich.net (198.108.22.137)	250.088 ms	250.429 ms	250.821 ms
9	ma.m-45.merit.edu (192.122.200.45)	250.247 ms	250.323 ms	250.756 ms
10	internet2.edu (207.75.164.248)	250.210 ms !X	250.327 ms !X	250.067 ms !X

To further illustrate, above is an example of the traceroute command on Ubuntu. The command was executed to the destination of 'perfsonar.net'. Each router is represented by the IP Address and DNS name, and the packets are sent to each router a total of three times, producing three different round trip time. Traceroute determines the latency time between the source and each intermediary notes and can greatly assist network engineers in finding the optimal path.

The tracepath command is similar to the traceroute command as it too can determine the path to the destination. However, the differences are that tracepath is able to determine that path to the destination is symmetrical, and it is able to determine the Maximum Transmission Unit (MTU). MTU determines the largest size of the packet or frame that can be sent on the network.

nazri @traceroute Analysis: ~$ tracepath perfsonar.net

1?	[LOCALHOST]	pmtu 1500
1	203.30.39.254	0.707ms
2	et-4-3-0.537.rtswlosa.net.internet2.edu	182.678ms
3	et-1-0-0.111.rtr.hous.net.internet2.edu	215.030ms
4	et-7-3-0.4070.rtsw.kans.net.internet2.edu	229.702ms
5	et-5-3-0.4070.rtsw.chic.net.internet2.edu	241.710ms
6	ae4x2058.chcg-lvl3-600w.mich.net	240.681ms
7	irbx69.anar-cor-cath.mich.net	250.670ms asymm 8
8	ae0x56.anar-cor-mitc.mich.net	250.497ms asymm 9
9	ma.m-45.merit.edu	250.674ms asymm 11
10	internet2.edu	250.993ms !H
Resume	pmtu 1500	

Above is an illustration of the tracepath command executed to the same host on Ubuntu. The tracepath command mapped the network path to the host, resulting in the same outcome as the traceroute command above. As mentioned, tracepath is able to highlight asymmetrical paths, as seen with the results flagged as "asymm".

With these tests, the DNS and round trip time are important as it provides the information that is required to troubleshoot the network. A lower round trip time is desirable as there is less

latency, signifying that the network is not congested. This interpretation of the data assist in steering the considerations in developing the web application. This is important as it allows the develop a web application that can display the data in a meaningful manner in the perspective of the network engineers.

2.1.2 Measurement of Latency (Round Trip Time)

Similar to traceroute and tracepath, ping is also a command available in Linux distribution and have been embedded as part of PerfSONAR. Ping measures the latency, specifically round trip time, from the source to the destination. Although all three commands are able to measure latency, ping only measures the latency to the destination as opposed to every intermediary node. Ping is also commonly used to troubleshoot the accessibility of a host.

```
nazri @traceroute-Analysis:~$ ping perfsonar.net -c 1
PING perfsonar.net (207.75.164.248) 56(84) bytes of data.
64 bytes from internet2.edu (207.75.164.248): icmp_seq=1 ttl=54 time=250 ms

--- perfsonar.net ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 250.453/250.453/250.453/0.000 ms
```

Above is an example of the ping command executed on Ubuntu. The ping command sends an ICMP “Echo” message to the host and waits for a ICMP “Echo Reply” to be sent back. The time it takes to receive the reply determines the round trip time. The above example shows that the round trip time to ‘perfonsar.net’ took 250.453 ms.

Similar to traceroute and tracepath, having an understanding of the data is important as it steers the design choices or consideration. It allows to build the web application in a meaningful manner for the network engineers. With regards to ping, a lower latency is desired as it signifies a non-congested network.

2.2 Deployment of PerfSONAR

PerfSONAR can be deployed in various manner. It can be deployed with all of the components, as discussed, onto a single host or with Esmond hosted on the other server.
The latter can also be considered as a deployment with a centralized measurement archive or a centralized Esmond server. Additional PerfSONAR host without Esmond can be connected to the centralized Esmond and use it to store the network metrics and data.

With regards to this project, Esmond is hosted on another server situated in Japan. As of now, only one PerfSONAR host is connected to the centralized Esmond server. This server is provided by SingAREN and is configured to run the network measurement tools. Future plans for this deployment includes adding additional PerfSONAR measurement hosts to the centralized Esmond to have a wider scope of network data. Having this understanding of the deployment structure is important as the results retrieved will be from different sources. This allows to steer the direction of the project, developing a web based application that can accommodate more than one measurement host.

A centralized Esmond host also reduces the complexity of the web based application as the data is queried from a single source as opposed to querying from multiple sources. Lastly,

this form of deployment also promotes separation of concerns between the various PerfSONAR hosts. This greatly simplify the way the hosts are managed and maintained.

Chapter 3 Review of Current Visualizations

Although the aim of the project is to build a web based application to visualize the data collected by PerfSONAR, there are other visualizations that have been built for PerfSONAR. This section of the report aims to look and review these current visualizations. Reviewing current visualizations are important as it also assist in steering the design choices and considerations in building the web application.

3.1 Golden Rules of a User Interface

Ben Shneiderman is an American Scientist who published the “Designing the User Interface: Strategies for Effective Human-Computer Interaction” book in 1985 [**12**]. His book is most known for the eight golden rules of a good interface design. These rules are widely discussed and prescribed to, including being taught to students in Nanyang Technological University as part of the Human Computer Interaction module. As such, the rules will be used to review existing visualizations. The eight golden rules of an interface design are as follows,

- Strive for consistency
  User interface should strive to be consistent. This includes the design of the individual items such as buttons, prompts, and the terminology used. The consistency allows users to gain familiarity.

- Enable expert users to use shortcuts
  This rule is aimed to recognize the diverse groups of users who will be using the application. It is specifically for frequent or expert users to use shortcuts to quickly get around with the application or accomplish the tasks.

- Offer informative feedback
  This rule emphasizes that for every action taken by the user, there should be a feedback in return. The size of the feedback should be dependent on the action the user has taken. Modest feedback can be inserted for minor actions such as clicking on an element of the application. Feedback is important to notify the user that an action has happened.

- Design dialogs to yield closure
  User Interfaces should provide a sense of closure to the user. Having a sense of closure to the user is important as it gives a sense of satisfaction or accomplishment. It is a signal to tell the user that this part of the application has finished, and the user can move on the other parts of the application. Applications without proper closure would make the users uncertain whether the series of tasks have been executed.

- Prevent errors
  A good user interface is able to prevent errors. This includes form validation preventing users from entering erroneous inputs. In an event that an error does occur, a good user interface should have a mechanism for the error to fail gracefully, and offer instructions for recovery.

- Allow reversal of action
  When it is possible, allow the application to do a reversal of action. This feature greatly relieves the anxiety of the user and promotes the user to explore unfamiliar parts of the application.

- Support internal locus of control
  A good user interface allows the user to be in control or be in charge of the user interface. It directly responds to the actions that the user has undertaken, and does not produce surprises or unfamiliar behavior.

- Reduce user's short-term memory load
  Humans have limited short-term memory capacity. According to a psychological study, the short-term memory capacity is limited to seven plus or minus two chucks of information [**13**]. With this, a good user interface should avoid situations where the users have to remember information from one aspect of the application to another.

Although these are considered as rules of a good user interface, it may not be applicable for every use case in an application. Nonetheless, these rules serves as heuristic in developing a good application for the users.

3.2 PerfSONAR Toolkit

The PerfSONAR toolkit comes with a graphical user interface (GUI) that aims to simplify the management and the configuration of the network measurement tests. It also provides an interface that allows the network engineers to visualize the network measurement results.

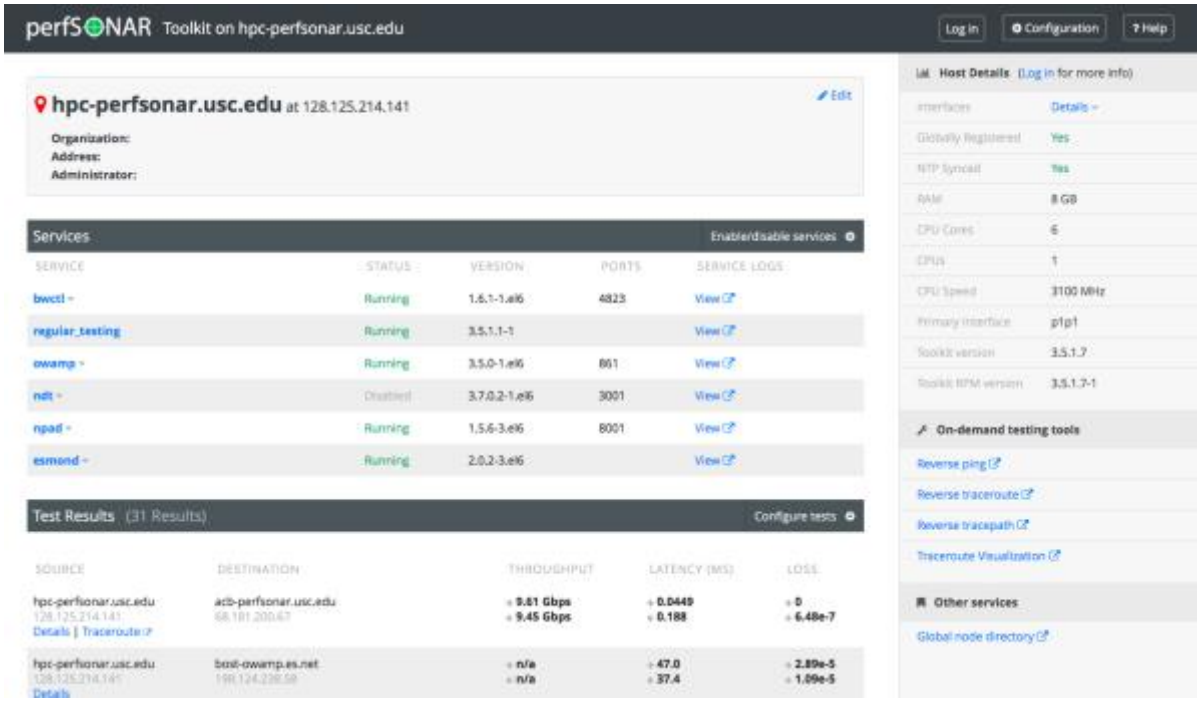


Figure 2. Screenshot of the main page in PerfSONAR GUI

The above figure shows the main page of the PerfSONAR GUI. Some of the golden rules previously discussed are present in this GUI. One of the rules is consistency. This is evident with the fonts used, the placement of the button, and the arrangement of the content on the site. An example is with the rows of test results, which are situated at the bottom half of the page. However, this particularly GUI raises issues as it was not designed in mind for the network engineers to observe the network in a grand scheme of things. With that, as a centralized Esmond server with multiple measurement hosts, the number of results will be substantially, making the page unnecessarily long. This can be seen with the screenshot below.

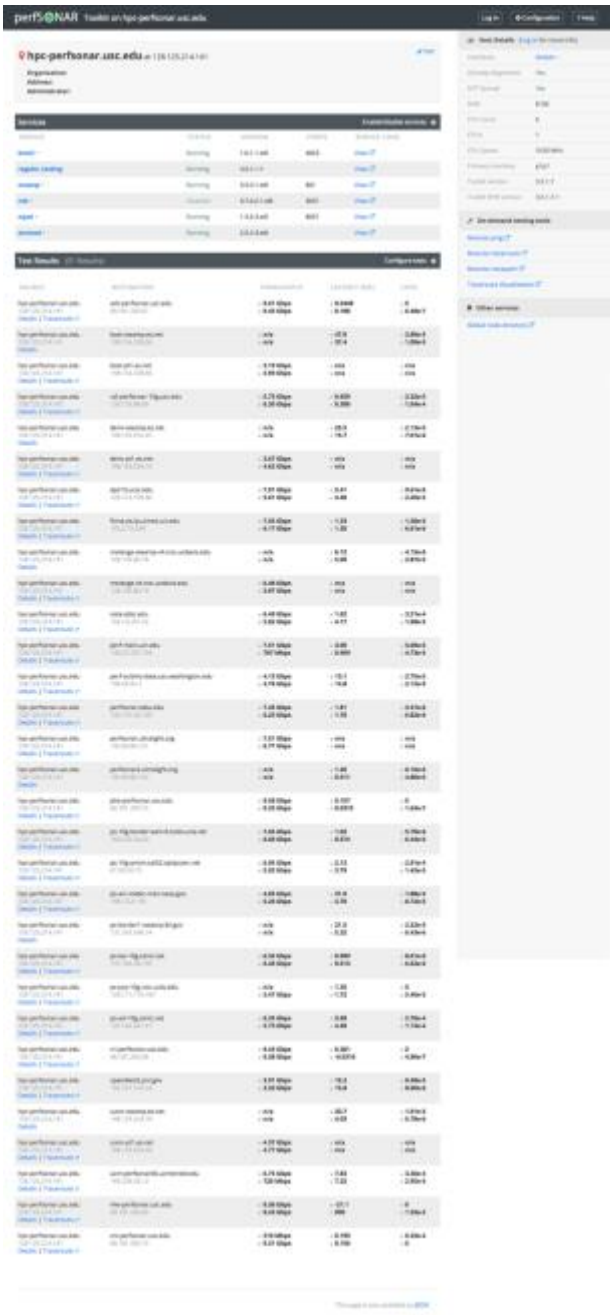


Figure 3. Full Screenshot of the PerfSONAR GUI.

This also raises usability issues as network engineers have to individually click on each results to view it. This violates the short-term memory load rule as the user will have to remember the content from the previous result and compare with the current result the user is viewing.

3.3 Monitoring and Debugging Dashboard (MaDDash)

The other visualization tool is Monitoring and Debugging Dashboard (MaDDash). Unlike the previous GUI which was created to observe the test results from a single instance of PerSONAR measurement host, MaDDash was created to observe network measurements from more than one instance of PerfSONAR [**14**].

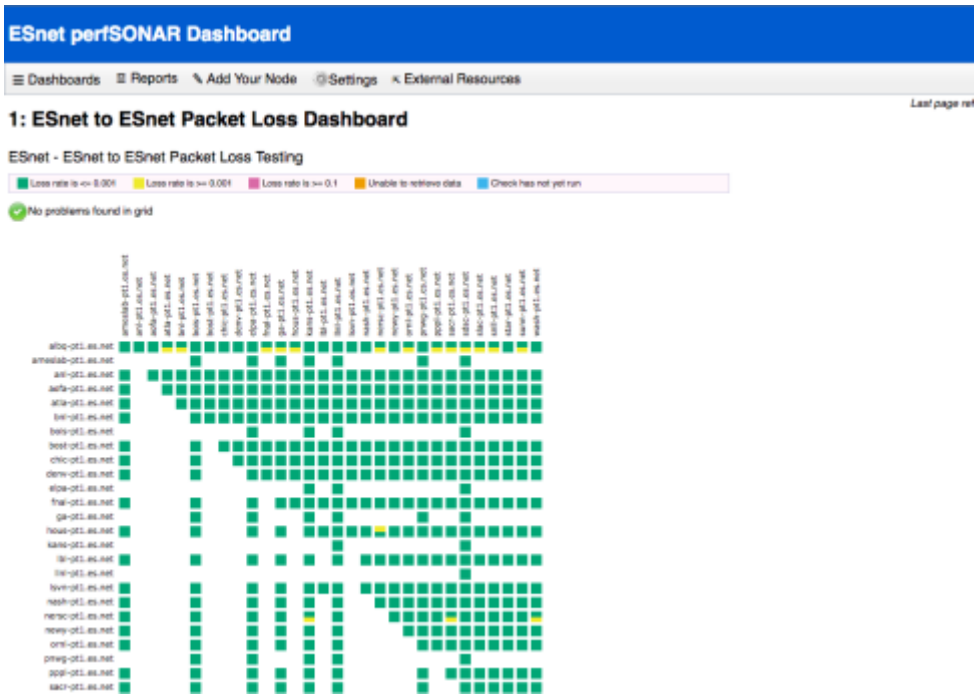


Figure 4. Screenshot of the MaDDash.

Above is a screenshot of MaDDash showing the network measurement results of various nodes. MaDDash is a well-designed GUI that abides by many of the eight golden rules of interfaces such as consistency in terms of design, reducing short-term memory load, and many more. However, MaDDash was designed for network measurements that are taken in a two-dimensional manner where each node listed has a measurement to all the other nodes thus making a grid structure.

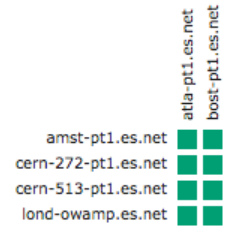


Figure 5. Screenshot of a smaller grid MaDDash.

To further illustrate, above is a small grid where the colors of the grid represents the status of the measurement. In this case, green represents a positive status. Each hosts listed has a measurement to all the other hosts. This grid visualization is ideal if the network measurement tests are configured in this manner. However, the grid will have a lot of white spaces or empty boxes if a centralized server was setup with the measurement hosts are not configured to have a network test to the same set hosts. A wide grid with empty spaces reduces usability for the user as the grid will be too large, causing users to scroll sideways incessantly.

Chapter 4 Front-End Development

The principal idea of the project is to develop a web based application with visualizations showcasing the various network metrics and to highlight networks with anomalies. The web based application should also provide ways for the users to view historical network metrics.
As such, the visualization is heavily interacted with the user, making usability paramount. This section of the report aims to cover the design and implementation in developing the front-end aspect. It also briefly covers the libraries and tools used.

4.1 AngularJS

The release of HTML5 and CSS3 has brought upon numerous features allowing developers to build web applications that rival against desktop applications. These modern web applications also utilize Asynchronous JavaScript and XML (AJAX) which gives web application the capability to send or receive new data without the page reloaded. Such web applications are classified as Single-Page Application. However, developing Single-Page Application poses challenges of its own such as the slow page load on the first load because the codes of Single-Page Application reside in one page and have to be loaded entirely. Single-Page Application also raises issues in development in terms of code maintainability and reusability.

AngularJS is a JavaScript open source framework released in 2010 by Google. Since it was released, usage of AngularJS has been growing steadily, and major companies such as PayPal have adopted AngularJS [**15**] [**16**]. It is a framework that adopts the Model View Controller (MVC) software design pattern which promotes separation of concerns where each layer of the application remains distinct, greatly promoting code maintainability and reusability.

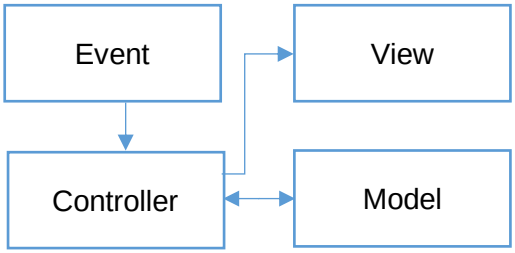


Figure 6. MVC Model

As the name suggest, the MVC design pattern defines individual layers as Models, Views and Controllers. The figure above graphically shows the structure of the MVC design pattern and how it relates to each of the different layers. Models are at the lowest layer of the pattern and are responsible for maintaining the state of the data. In AngularJS, these models are represented as JavaScript Objects. Views are the User Interface layers which are interacted by the users. These are the HTML files with the various User Interface components such as fields and textboxes. Controllers plays an important role in connecting the view and the model. It responds to the users' request or events and perform interactions as needed such as modifying the state of the model objects or redirecting to another HTML page.

4.1.1 Services

The MVC pattern brought separation of concerns which are important in promoting code reusability and maintainability. In AngularJS, further separation of concerns can be done through services. Services are similar to global functions where it can be called in any controllers or to be used to any forms of data such as on any of the models.

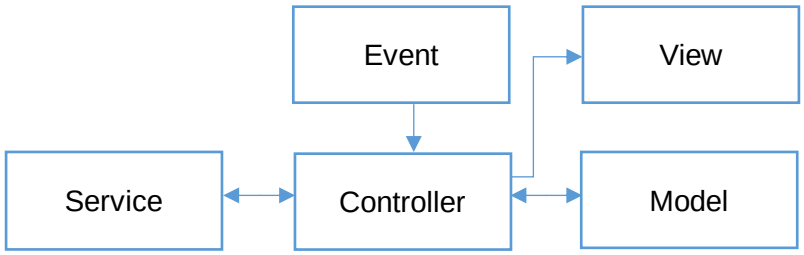


Figure 7. MVC Model with Services in AngularJS

The diagram above shows the updated design pattern with services. These services can be used for different types of data as opposed to controllers where the business logic that resides are more specific to the model it interacts with. Additionally, in AngularJS, only a single instance of the service is instantiated providing a mechanism for Controllers to share data if needed.

4.2 Software Architecture

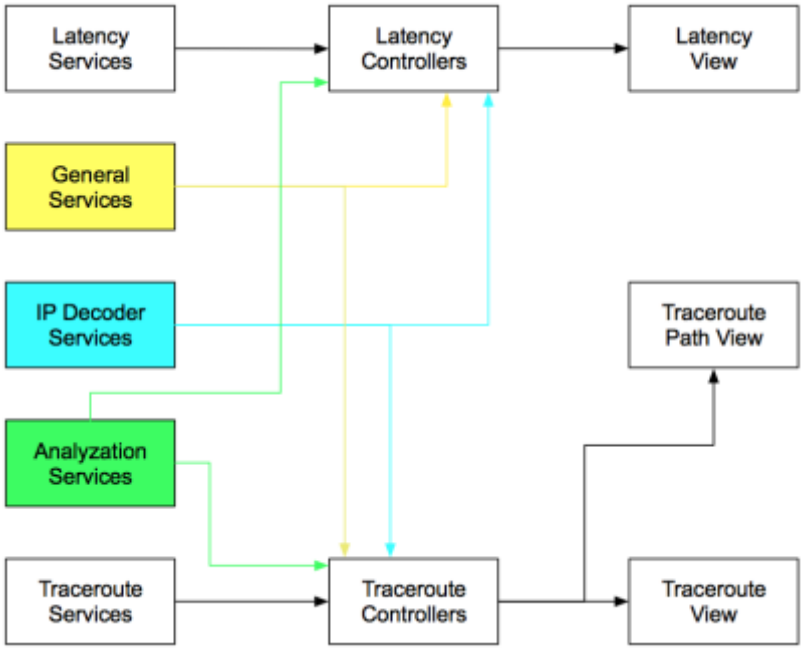


Figure 8. High Level Architecture. Services are color coded for easy viewing.

This section covers the software architecture of the web-based application for the front-end development of the project. There are many controllers and services that have been defined for front-end aspect of the web based application. However, these services or controllers are similar in terms of the purpose it possess. As such, the figure above shows the high level

architecture of the front-end development, and the categories of types of controllers or services that exist. These services and controllers will be discussed in the following section.

4.2.1 Latency Controllers

As discussed, controllers are the intermediary entities between views and models. However, there are no models in this architecture as the web based application retrieves the data directly from Esmond and build the visualization. The Latency Controllers contains multiple sub controllers that are primarily focused on building and managing the visualization for the latency dataset. This aspect includes adding nodes, adding edges, controlling the visualization layout to avoid collision, and color coding the nodes and edges that are abnormal. Determining abnormality is done through the Analyzation Services that scans the data and returns a result back to Latency Controllers.

4.2.2 Latency Services

The Latency Services contains global functions to be used only with the controllers defined under Latency Controllers. One of the services declared is called LatencyResultsService. This service utilizes the REST interface in Esmond to pull specifically latency related data. Having this as a service allows the controllers to pull data as needed compared to each controller having its own implementation to utilize the REST interface.

4.2.3 Traceroute Controllers

Similar to Latency Controllers, there are multiple sub controllers within this category that are primarily focused on building and managing the visualization for the traceroute dataset. This service is also utilized the Analyzation Services to determine if the round trip time values are above the Upper Fence.

4.2.4 Traceroute Services

The Traceroute Services are akin to the Latency Services. It contains the global functions that is to be used only by the controllers that defined under Traceroute Controllers. It contains a similar function called TransrouteResultService that will utilize the REST interface in Esmond to pull traceroute related data.

4.2.5 General Services

As the name suggest, General Services contains all of the services that are for general use. This service is used by both Traceroute and Latency Controllers. The services declared are,

- UnixTimeConverterService
  A service that coverts the time into Unix format, or also known as POSIX or Epoch format.

- CurrentTimeUnixService
  A service that returns the current time into Unix form, also kown as POSIX or Epoch format.

- HostService
  A service that returns the IP address of the centralized Esmond. This promotes easy maintainability as the IP address is not littered in all of the files.

- UniqueArrayService
  A service that takes in an array and returns back an array with only unique values. This is used determine unique nodes to add into the visualization.

4.2.6 IP Decoder Services

This service can be used by both Latency and Traceroute Controllers. It contains the following services,

- GeoIPNekudoService
  This service retrieves the Country and City of an IP Address.

- DNSLookup
  This service executes a reverse DNS lookup, returning the DNS of an IP Address.

Both of these service aims to improve the usability of the network engineers. This is done through providing additional information of the IP Address such as country, city and DNS.

4.2.7 Analyzation Services

The Analyzation Services are one of the important services as it determines if the dataset retrieved are above the threshold as defined by the Upper Fence. It contains three separate services,

- AnalyzeTracerouteRtt
  This service takes in the traceroute data, calculates the Upper Fence and flag results that are above the threshold.

- AnalyzeTraceroutePath
  This service takes in traceroute data and check for any changes in path.

- AnalyzeLatency
  This services takes in latency data and check for abnormalities.

The Analyzation Services utilized the MathJS library, a JavaScript library that contains various mathematical functions.

4.3 Implementation

On developing the front-end aspect of the project, there are a few fundamental tasks to understand. These are,

- Retrieving data through REST, manipulating the data if needed
- Creating the visualization with the data
- Caching the dataset for optimization and further use

These will be further discussed below.

4.3.1 Retrieving data through REST

As previously discussed, Esmond in PerfSONAR provides a REST interface to retrieve the network measurement tests. The other benefits of using AngularJS is to tap on the libraries and tools such as the $http service that is able to utilize the REST interface.

```
// Codes above omitted

$http({
  method: 'GET',
  url: HostService.getHost(),
  params: {
   'format': 'json',
   'event-type': 'packet-trace',
  //604800 seconds = 7 days
   'time-range': 604800
  }
}).then(function(response) {
  //Successfully pulled data
  //Success Callback

}, function(response) {
  //Failed to pulled data due to server error
  //Failed Callback


});

// Codes below omitted
```

Above is an example of the $http service retrieving the data through the REST interface. In this example, a HTTP GET request is executed with the URL specific through another user defined service, HostService. HostService is a service that returns the address of Esmond. HTTP is a protocol that allows communication between the client's browser and the server. With regards to this example, $http service is executing a HTTP GET request to the desired Esmond server, specifically requesting the data.

Parameters can also be provided with the HTTP GET request, this is evident with the '**params**' clause specified. This clause specifies the query string in an URL which allows to send additional information to the server. The parameters given above translates to the following query strings,

```
http://URL.com/?format=json&event-type=packet-trace&time-range=604800
```

The parameters defined is requesting for data with the event type as packet trace, from the last seven days and in the format of JavaScript Object Notation (JSON). Query strings are important as it allows to be specific or fine-tune the data to retrieve.

JavaScript Object Notation (JSON) is an open standard format to transmit information between the client and server. It is an independent data format and can be used in any programming language, similar to XML. If the HTTP GET request is success, a variable containing the JSON will be given and can be parsed.


```
{
  "city": false,
  "country": {
  "name": "United States",
    "code": "US"
},
  "location": {
  "accuracy_radius": 1000,
    "latitude": 37.751,
    "longitude": -97.822
},
  "ip": "8.8.8.8"
}
```


Above is an example of a JSON response from http://geoip.nekudo.com/api/8.8.8.8, one of the web services used in the project to geocode IP addresses. JSON format arrange the data in a key and value pair and can be easily retrieved with the following syntax,

```
var city = response.data.city;
// city = false
//response is from the success call back defined above
```

Once the data is received and parsed, additional logic such as conducting analysis to determine if an anomaly is present can be conducted. This project has an Analyzation Service that can be called to conduct the analysis on the data and the Analyzation Service will return a Boolean if an anomaly exists.

The $http service is important as it is dominantly used to communicate with all servers including the back-end server, the Esmond server, and the IP address geocoding web service as shown above.

4.3.2 Cytoscape.js

Cytoscape.js (Cytoscape) is an open source JavaScript network visualization library [**17**]. It provides the tools necessary to build and draw an interactive visualization and to anchor events onto the visualization, allowing users to interact with the visualization. Cytoscape is specifically used after retrieving the data from PerfSONAR through the $http service.

```
var cy = cytoscape({ /* omitted for clarity */ });

// Adding Nodes
cy.add([{
```

```
    group: "nodes",
    data: {
      id: "59.43.250.106",
      country:"China"
    }

  }, {
    group: "nodes",
    data: {
      id: "59.43.250.117",
      country:"China"
    }
  }]);

  // Adding Edges
  cy.add({
    group: "edges",
    data: {
      id: "edge0",
      source: "59.43.250.106",
      target: "59.43.250.117",
      latency: 123
    }
  });
```

The code snippet above shows the usage of Cytoscape. The visualization first has to be declared. From the snippet above, this is done with the variable `cy`. Upon declaring the variable, nodes can be added onto the visualization. Cytoscape allows the addition of nodes in bulk, specifically though an array, this is especially useful as it helps to expedite the process of building the visualization. Upon adding the nodes, an edge can be added between the two nodes with the correct ID specified. The ID field is a mandatory field in Cytoscape.

One of the advantages of using Cytoscape is additional information can be inserted into the node or edges. This greatly reduce complexity in building the visualization in passing information within the web based application. The example above shows this capability with the fields "**country**" in the nodes and "**latency**" in the edge which are user defined. Cytoscape provides the capability to dynamically update and show this values on the visualization.

4.3.3 Cache Implementation

In computing, cache is defined as temporary storage to store data to expedite future requests. With regards to cache in this project, the browser cache is greatly utilized to optimize the speed in building the visualization. The cache that has been leveraged is based on the HTML5 Web Storage API, specifically Local Storage which remains persistent even when the browser is closed [**18**]. A wrapper library called Angular Cache was used to simplify the usage of the HTML5 Web Storage API.

```
var DNSCache;

if (!CacheFactory.get("ReverseDNSLookup")) {
  // Cache not found, create new cache.
   DNSCache = CacheFactory("ReverseDNSLookup", {
    // Items added to this cache expire after 5 days, time in milliseconds.
     maxAge: 7200 * 60 * 1000,

    // Items will be deleted from this cache right when they expire.
    deleteOnExpire: 'aggressive',

    // This cache uses localStorage
    storageMode: 'localStorage'
  });
}
// Storing into cache. Key and Value format.
DNSCache.put('8.8.8.8', 'google-public-dns-a.google.com');
```

Above is a code snippet of the cache implementation for reverse DNS lookup. The code above creates a cache using the HTML 5 Local Storage API, stores it for five days, and will be deleted upon expiration. Caching the the results of the reverse DNS lookup is ideal as this reduce the overload to continuously send a HTTP GET request to get the results of the reverse DNS Lookup. Other than DNS cache, the IP Geocoding is also cached, this also improves the performance and serve as a prevention from hitting the API limit of the web service.

4.4 User Interface

The User Interface is a paramount aspect of the web application as it serves as an entry point for the network engineers to use the visualizations. It is important to incorporate the design choices and considerations in order to create an informative or meaningful visualization. These design considerations include the eight golden rules of user interface and the interpretation of the network measurement data that was discussed above.

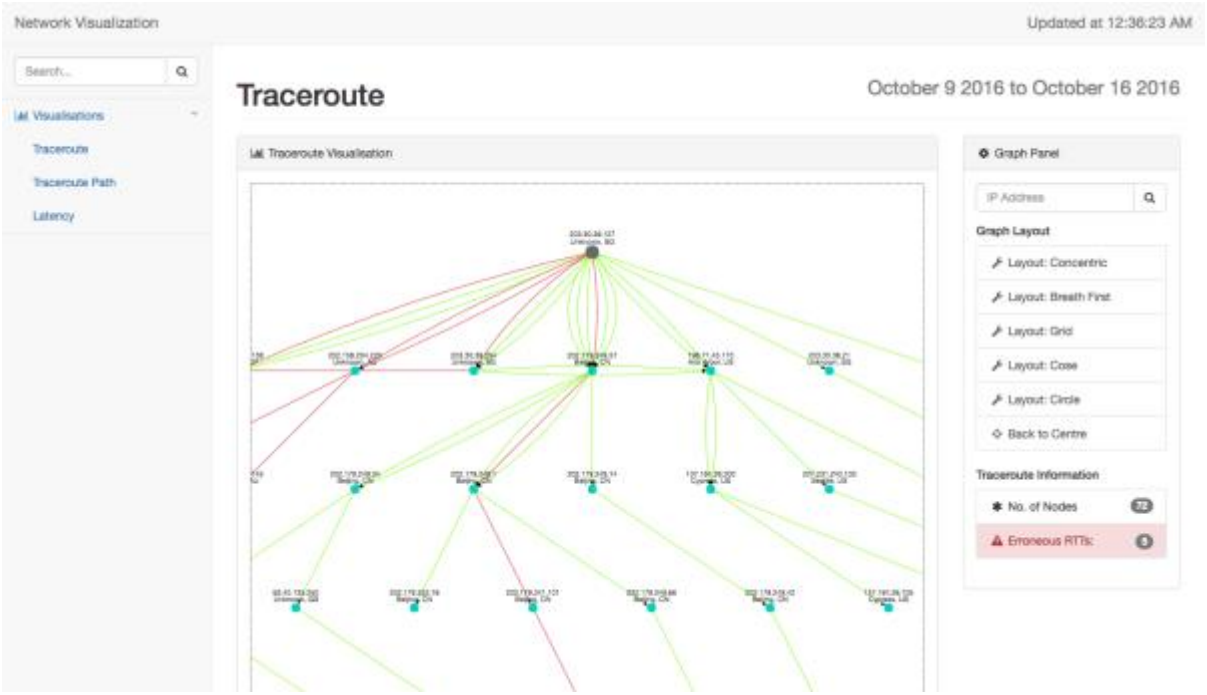


Figure 9. Screenshot of Traceroute visualization.

Above is a screenshot of the one of the few visualizations in the web based application. With the visualization taking the majority of the space, network engineers are able to have an overview of all the network measurement results. Additional controls are given at the right side of the page allowing the network engineers to change the layout of the nodes and edges. The graph is also interactive providing further controls such as to zoom, pan across the visualization, and to manually move the nodes and edges if needed. Most importantly, as the number of nodes increases, a search bar has been implemented to ease in locating the correct nodes.

The integrity of the data is kept by creating a visualization that resembles the data. This allows the visualization to be informative and meaningful for the network engineer. An example is shown in Figure 9 with the traceroute visualization. The link to the various intermediary nodes are shown along with colored edges which determines the status of the latency.

To access additional visualizations, this can be done with the panel on the left side of the page. As noted, the other visualizations are Traceroute Path and Latency. The other visualizations adopt the same layout, promoting consistency within the web application, in line with the eight golden rules of user interface. Screenshots of the other visualizations is attached at the appendix.

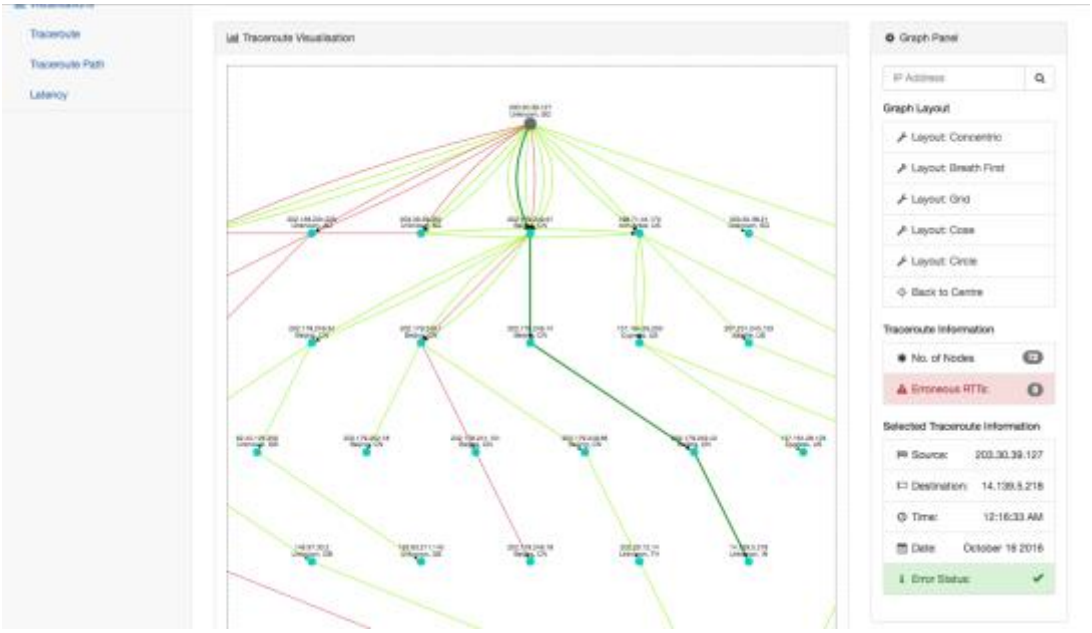


Figure 10. Screenshot with additional information.

Additional information is also shown at the panel on the right upon interacting with the visualization, shown on the bottom right side of Figure 10. The interface also utilizes the other golden interface rules, offering informative feedback. This is evident with the path turning a darker color as it is clicked, indicating a selected path.

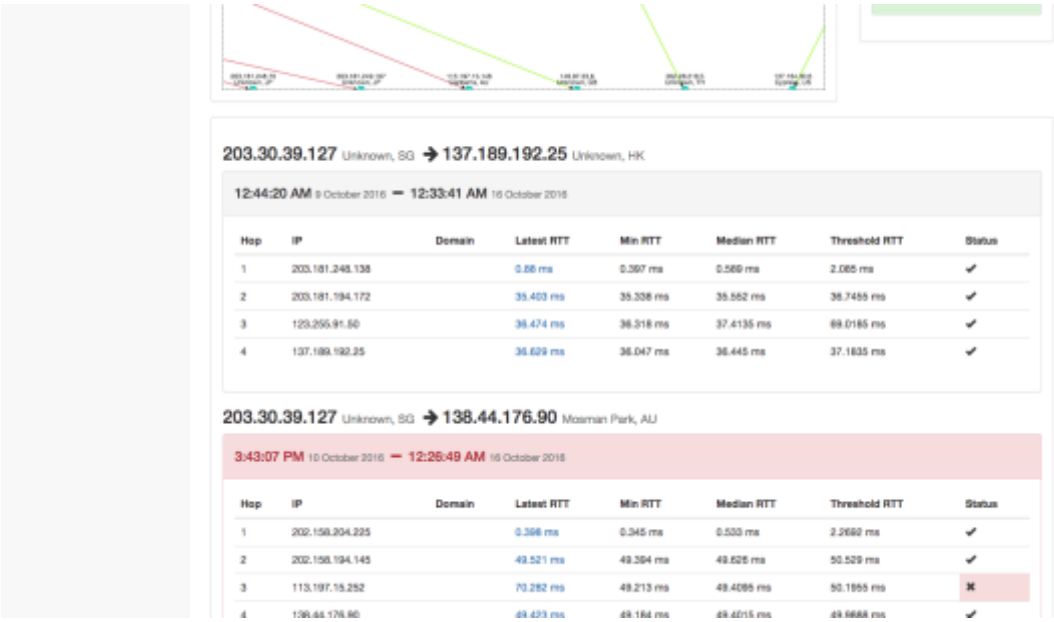


Figure 11. Screenshot with additional information.

Further information can be found once the user clicks on the “Error Status” at the right side of the panel as shown in Figure 10. This will redirect the user to the correct table underneath the visualization that shows the necessary information such as the round trip values.

Chapter 5 Back-End Development

The back-end development or also commonly known as server-side development is primarily focused on supporting the visualization built in the front-end development phase. The back-end aspect of the project is able to provide additional support through providing features such as persistent storage and to handle complex or computationally heavy business logic.

The primary method of interaction between the front-end and back-end is through AJAX requests. AJAX requests allows web applications to be interactive as new data can be loaded without reloading the page.

5.1 Flask

Similar to front-end development, the back-end development utilizes a back-end web application framework. The use of frameworks provides access to common functionalities and interfaces that have been pre-built and tested thoroughly.

Flask is a python based micro web application framework. It aims to be a minimal and extensible framework thus only contain the core libraries of a web application. It does not contain additional libraries such as data abstraction layers or server-side form validation [**19**]. The extensible nature of Flask makes it an ideal and flexible framework as it can greatly enhance its set of functionalities to support future enhancement.

The use of Python for back-end development is an ideal choice. Python is widely supported in numerous operating systems thus making it a versatile programming language. This versatility is an advantage as this lax the requirements needed for deployment. Additionally, a survey conducted by data professionals concluded that Python is the preferred language when it comes to data science because of the wealth of third-party libraries including machine learning [**20**]. This again makes Python an ideal programming language as relevant data analytics libraries can be tapped to conduct analysis on the network.

As Flask is a micro web application framework, it can be run entirely out of one Python file.

F546_Flask.py

```
# ...import statements excluded.

@app.route('/helloworld')
def helloworld():
    print "Hello World"

if __name__ == '__main__':
    app.run(debug=True, port=8000)
```

The above is a code snippet demonstrating the usage of Flask. As the user visit the URL, http://localhost:8000/helloworld, the function `helloworld()` is executed thus printing "`Hello World`".

F546_Flask.py

```
# ..import statements excluded

@app.route('/helloyou)
def helloyou():
    name = request.args.get('name')
    print "Hello " + name

if __name__ =='__main__':
   app.run(debug=True, port=8000)
```

The above is a method to pass information from the front-end onto the back-end server. This is done through query string. Query strings are an important aspect of URL and have been discussed in Chapter 4. To illustrate the usage of query string, the following is an URL to be executed, http://localhost:8000/helloyou?name=John

The URL executed will run the helloyou() function and it will search for the query string “name”, retrieving the value of John.

5.2 Implementation

Flask was used for two main reasons. The first reason was to use as a redirection tool. Below is the implementation shown in flask. When a user visits the URL website, it is automatically redirect to the index HTML page where the visualization begins.

F546_Flask.py

```
# ..import statements omitted

@app.route('/')
def index():
    return app.send_static_file('index.html')

# ...Remaining codes omitted
```

Another implementation of Flask was to use reverse DNS lookup in Python. Unfortunately, JavaScript does not have a built in function to execute reverse DNS lookup as such this was done in the backend.

F546_Flask.py

```
# ..import statements omitted

@app.route('/reversednslookup')
def reversedns():
    try:
        ipaddr = request.args.get('ipaddress')
        resolver = dns.resolver.Resolver()

        addr = dns.reversename.from_address(ipaddr)
        toPrint = str(resolver.query(addr,"PTR")[0])
        return jsonify(dns=toPrint, ip=ipaddr)
    except dns.exception.SyntaxError:
        return jsonify(dns="Syntax Error", ip=ipaddr)
    except dns.exception.Timeout:
        return jsonify(dns="Request Timed Out",
ip=ipaddr)
    except dns.exception.DNSException:
        return jsonify(dns="Unknown", ip=ipaddr)
    except dns.exception.TooBig:
        return jsonify(dns="Unknown", ip=ipaddr)
    except dns.exception:
        return jsonify(dns="Unknown", ip=ipaddr)

# … Remaining codes omitted.
```

The front-end can request for reverse lookup with the following URL, http://localhost/reversednslookup?ipaddress=8.8.8.8

Flask application will retrieve the IP Address as 8.8.8.8 and continue with reverse DNS. It will print out the result as a JavaScript Object Notation (JSON).

```
{
  "dns": "google-public-dns-a.google.com.",
  "ip": "8.8.8.8"
}
```

With JSON, the front-end is able to retrieve and access it as a JavaScript object. This aspect has been discussed in Chapter 4.

Chapter 6 Deployment

In order to make the visualization available to the users, it requires to be deployed onto a server. The two main packages required as part of deployment are Python and Nginx, both of which are widely supported in various operating systems including Linux, OS X and Windows. This greatly increases the flexibility of the operating system for deployment.

A deployment server has been provided kindly by SingAREN. The installed operating system is Ubuntu 16.04.1 LTS, one of the many available Linux distributions. It is a virtualized machine with the following specification,

- Intel(R) Xeon(TM) CPU 3.00GHz - 32 Bits with 4 Logical CPU
- System memory of 1019 MB

6.1 Deployment Guide

This section of the report covers the steps in deploying the visualization. A user with root access is required as this provides the user with sufficient rights and permissions to install and configure the necessary packages.

6.1.1 Installing dependencies

The following dependencies are required,

- python-dev
- nginx
- python-pip

To install the dependencies on Ubuntu, execute the following command,

```
sudo apt-get install python-pip python-dev nginx
```

The ‘apt-get’ command is part of package manager list of commands on Ubuntu. Package managers are software that handles installation and removal of packages for the operating system.

6.1.2 Creating a Python Virtual Environment

A Python-based virtual environment allows the web application to reside in isolation. This prevents other users of the operating system to carelessly update or remove packages that are crucial to the visualization. In order to create the virtual environment, the package first has to be installed using the following command,

```
sudo pip install virtualenv
```

Similar to `apt-get`, `pip` is a package manager solely to manage Python-based packages. It has the same functionalities such as installing and removing packages. With the virtual environment package installed, a virtual environment can be created with the following command,

```
sudo virtualenv /var/F546Backend/VirtualEnvr_F546
```

With the virtual environment created, packages for the visualization can proceed to be installed in the virtual environment. In order to install packages inside the virtual environment, the following command is needed to activate it,

```
source /var/F546Backend/VirtualEnvr_F546/bin/activate
```

The prompt will change with the name of the virtual environment in parenthesis at the front. This is to indicate the user is in the virtual environment. The following is an example.

```
(VirtualEnvr_F546) nazri@tracerouteAnalysis:/var/F546Backend$
```

With the virtual environment activated, proceed on to install the dependencies required for the visualization. This can be done with the following command,

```
pip install uwsgi flask
```

The above command requests pip to install Flask and uWSGI on the host machine. The Flask framework is needed as it contains the necessary libraries to serve the web application. uWSGI is a package that provides the Web Server Gateway Interface (WGSI) protocol that will be used along with Nginx to serve the web application.

Exit the virtual environment with the following command and proceed on with the remaining steps. The name of the virtual environment will no longer be in parenthesis at the beginning of the prompt.

```
deactivate
```

6.1.3 Setting up the project folder

With the necessary packages installed, configurations are needed to proceed with the deployment. Create the following folder,

- /var/F546Backend/F546_Flask
  This python file containing the Flask codes resides in this directory.

- /var/F546Backend/F546_Flask/static
  The HTML files from the Front-End development resides in this directory.

6.1.3.1 Creating the WSGI file

In /var/F546Backend/F546_Flask, create the wgsi.py file. The file contains the following,

/var/F546Backend/F546_Flask/wsgi.py

```
from FlaskApp import app

if __name__ == "__main__":
    app.run()
```

The above file serves as an entry point for WSGI, telling WSGI to run the Flask application.

Additionally, in the same directory, create the FlaskApp.ini file. The file contains the following,

/var/F546Backend/F546_Flask/FlaskApp.ini

```
[uwsgi]
module = wsgi:app

master = true
processes = 4

socket = F546App.sock
chmod-socket = 660
vacuum = true

die-on-term = true
```

The above file serves as a static configuration file for WSGI.

6.1.4 Configuring automated start using systemd

In Linux, systemd is a service that allows automatic startup of processes. Configuring systemd with the web application allows the WSGI protocol to automatically start when the server reboots. In order to configure this, a configuration file is created,

/etc/systemd/system/F546App.service

```
[Unit]
Description=uWSGI for PerfSONAR Data Visualization
After=network.target

[Service]
User=root
Group=www-data
WorkingDirectory=/var/F546Backend/F546_Flask
Environment="PATH=/var/F546Backend/Virtual Envr_F546/bin"
ExecStart=/var/F546Backend/Virtual Envr_F546/bin/uwsgi --ini
FlaskApp.ini

[Install]
WantedBy=multi-user.target
```

With the file F546App.service saved, the following commands should be executed. It will register the the new configuration file and startup WSGI.

```
sudo systemctl start F546App
sudo systemctl enable F546App
```

6.1.5 Configuring Nginx to serve the web application

This step in the deployment involves configuring Nginx to serve the web requests. The web requests received in Nginx will be passed to the web application using the WSGI protocol. To begin, a configuration file is needed. This configuration file is created as follow,

/etx/nginx/sites-available/F546App

```
server {
    listen 80;
     server_name 203.30.39.133;

    location / {
        include uwsgi_params;
         uwsgi_pass
unix:///var/F546Backend/F546_Flask/F546App.sock;
    }
}
```

The 'F546App.sock' name correspond with the name previously set in the FlaskApp.ini file above. Upon creating the file, a link to the file is needed to in the directory, /etc/nginx/sites-enabled to enable it. This can be done with the following command,

```
sudo ln -s /etc/nginx/sites-available/F546App /etc/nginx/sites-enabled
```

Additionally, in the same directory, the modify the /etc/nginx/sites-enabled/default and append the following in the server clause, with the URL pointing to the Esmond server.

/etx/nginx/sites-available/default

```
server {
  # ...(Full Code Listing in Appendix)
  listen 8080 default_server;
  listen [::]:8080 default_server;
   server_name localhost;
    more_set_headers 'Access-Control-Allow-Origin: *';

  location /cors/ {
http://ps2.jp.apan.net/esmond/perfsonar/archive/;
  }
}
```

The following command is an optional command. It tests the nginx configuration files for error,

```
sudo nginx -t
```

If there are no issues, the Nginx process should be restarted to read in the new configuration file, this can be done by,

```
sudo systemctl restart nginx
```

Lastly, it is important to open the ports for Nginx to accept the web requests. This can be done with the following command,

```
sudo ufw allow 'Nginx Full'
```

Chapter 7 Conclusion

With networks becoming pervasive and an integral part of society, network measurement tools such as PerfSONAR are becoming increasingly important. These tools empower network engineers by providing valuable insights of the network, allowing to assist in determining failures and meeting expectations of users.

PerSONAR is a powerful tool that is able to capture a wide variety of network metrics. The architecture of PerfSONAR allows it to be extensible as new tools or services can be added as part of future enhancement. Additionally, with the REST interface in PerfSONAR provided by Esmond, it encourages the development of third party tools to further extend the usage of PerfSONAR. These third party tools are able to further make PerfSONAR a valuable toolkit for network engineers.

There are a couple of visualizations or GUI built for PerfSONAR. The GUI provided in PerfSONAR is an excellent tool as it reduces the complexity in using PerfSONAR. However, it is not an ideal tool for network engineers to observe the network, especially in a grand scheme of things.

The web based application built in this project aims to fill this gap, providing network engineers the ability to observe the network from a larger perspective. The web based application utilized reliable libraries such as AngularJS for the front-end development and Flask for the back-end development. The use of frameworks simplifies the development and tap on the included libraries or functionalities to build a modern web application. More importantly, the use of framework promotes code reusability and maintainability allowing opportunities for further enhancement, especially in regards of the back-end development where machine learning libraries can be utilized to enhance the analyzation aspect. This is ideal as it allows the web application to continue being an important tool for PerfSONAR.

References

[1] Internet Society. Brief History of the Internet. [Online]. http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet

[2] CERN. The Large Hadron Collider. [Online]. http://home.cern/topics/large-hadron-collider

[3] CERN. About | Worldwide LHC Computing Grid. [Online]. http://wlcg-public.web.cern.ch/about

[4] Rohini Prinja, Abhishek Chandra, Zhi-Li Zhang Sourabh Jain. (2008, April) Failure Classification and Inference in Large-Scale Systems: A Systematic Study of Failures in PlanetLab. [Online]. https://www.dtc.umn.edu/publications/reports/2008_08.e

[5] ESnet. PerfSONAR. [Online]. http://www.perfsonar.net/about/what-is-perfsonar/

[6] Jeff W. Boote, Eric L. Boyd, Jérôme Durand, Loukik Kudarimoti, Roman Łapacz, D. Martin Swany, Szymon Trocha, Jason Zurawski Andreas Hanemann. (2005) PerfSONAR: A Service Oriented Architecture for Multi-domain Network Monitoring. [Online]. https://www.es.net/assets/pubs_presos/hbbd05.pdf

[7] PerfSONAR. (2016, May) The perfSONAR Measurement Framework: Project Update and Roadmap. [Online]. https://meetings.internet2.edu/media/medialibrary/2016/06/09/20160517-boyd-perfSONAR.pdf

[8] ESnet. esmond: ESnet Monitoring Daemon. [Online]. http://software.es.net/esmond/perfsonar_gridftp.html

[9] PerfSONAR. PerfSONAR User Guide. [Online]. http://docs.perfsonar.net/manage_regular_tests.html

[11] Cisco. (2006, November) Understanding the Ping and Traceroute Commands. [Online]. http://www.cisco.com/c/en/us/support/docs/ios-nx-os-software/ios-software-releases-121-mainline/12778-ping-traceroute.html

[10] Richard A Steenbergen. A Practical Guide to (Correctly) troubleshooting with Traceroute. [Online]. https://www.nanog.org/meetings/nanog47/presentations/Sunday/RAS_Traceroute_N47_Sun.pdf

[12] Ben Shneiderman. University of Maryland. [Online]. https://www.cs.umd.edu/users/ben/goldenrules.html

[13] George A. Miller, "The Magical Number Seven, Plus or Minus Two: Some Limits on our Capacity for Processing Information," Psychological Review, vol. 63, pp. 81-97, 1956.

[14] ESnet. MaDDash: Monitoring and Debugging Dashboard. [Online]. http://software.es.net/maddash/

[15] BuiltWith. BuiltWith Technology Lookup. [Online]. http://trends.builtwith.com/javascript/Angular-JS

[16] Daniel Brain. (2015, September) Daniel Brain - Medium. [Online]. https://medium.com/@bluepnume/sane-scalable-angular-apps-are-tricky-but-not-impossible-lessons-learned-from-paypal-checkout-c5320558d4ef#.emldc0syy

[17] Lopes CT, Huck G, Dong Y, Sumer O, Bader GD Franz M. Cytoscape.js. [Online]. http://js.cytoscape.org/

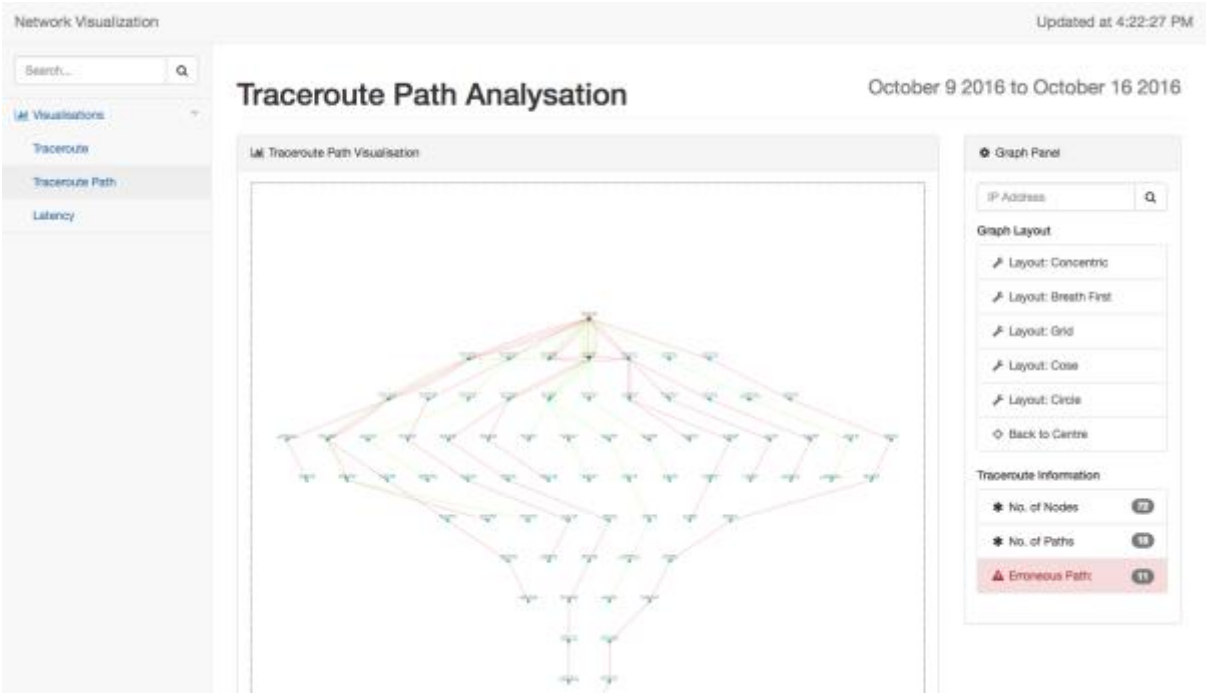
[18] Mozilla Developer Network. Web APIs. [Online]. https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API

[19] Armin Ronacher. Flask - A Python Microframework. [Online]. http://flask.pocoo.org/docs/0.10/foreword/#

[20] Packt, Big Data and BI: Salary and Skills Report, Richard Gall, Ed. United Kingdom of Great Britain and Northern Ireland: Packt Publishing, 2015.

Appendices

Traceroute Path Visualization Interface



Latency Visualization Interface

/etc/nginx/sites-available/default

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;

        root /var/www/html;

        index index.html index.htm index.nginx-debian.html;
        server_name _;
        location / {
                try_files $uri $uri/ =404;
        }

  listen 8080 default_server;
  listen [::]:8080 default_server;
  server_name localhost;

  more_set_headers 'Access-Control-Allow-Origin: *';

  location /cors/  {
    proxy_pass http://ps2.jp.apan.net/esmond/perfsonar/archive/;
  }
}
```