EVALUATING HTTP PERFORMANCE FROM STREAMS

DESIGN DOCUMENT

- **Team Name:** NAGIOS

- **Team Members:**

  - Atla Prashant
  - Chilukuri, Megh Phani Dutt
  - Garg, Prafull
  - Grandhi, Veera Venkata Santosh S G
  - Kalidindi, Rajeev Varma
  - Kolli, Samuel Sushanth
  - Madala, Sravya
  - Musinada, Suren
  - Naguru, Sriram Prashanth
  - Peddireddy, Divya
  - Rajana, Poojitha

- **Document Type:** Design Document

- **Version Number:** Version 1.4

- **Publication Date:** August 24th, 2015

1. PREFACE:

This project is concerned with evaluating HTTP performance from streams. We describe the version release v1.4 of the design document, where we elucidate on how to develop a tool to monitor the HTTP traffic in a typical data center.

The document is partitioned into various sections. Section 2 gives an overview of the abbreviations used in the document. Sections 3, 4 and 5 describe the back end, front end and the REST API respectively. The last section suggests the references used to prepare this document.

Release v1.4 on 2015-08-24

- Updated release

Version history is as follows:

PUBLICATION DATE	VERSION	DESCRIPTION	CHANGES
2015-08-24	v1.4	Updated version	Updated the following from the feedback of the CEO: • Provided detailed description for test descriptions, especially operations and expected result. • Importing data from a third party via a REST API.
2015-06-01	v1.3	Updated version	Updated the following from the feedback of the CEO: • Updated performance metrics test • MySql database interaction with the frontend • RESTful API
2015-05-20	v1.2	Updated version	Updated the following from feedback of the CEO • MySql database interaction • RESTful API
2015-05-14	v1.1	Updated version	Updated the following from feedback of the CEO • Detailed design • Unit test plans • RESTful API
2015-05-05	v1.0	Initial release	

2. GLOSSARY AND ABBREVIATIONS:

HTTP: Hypertext Transfer Protocol

It is a protocol at the application level for communication of data between the network elements such as clients and servers.

GUI: Graphical User Interface

An interface which allows the users to communicate with the electronic devices through visual icons. In some cases, it contains audio feedback as well as voice control.

DPMI: Distributed Passive Measurement Infrastructure

This interface is used to read the data stream at various measuring points.

RESTful: Representational State Transfer

An architectural pattern to improve portability, scalability of the system.

API: Application Programming Interface

This specifies how software components should interact with each other.

3. MODULE 1: BACKEND

The purpose of this module is to do calculations based on the data from the DPMI. Its position is between the DPMI and the front end in the high level architecture. It interfaces with the DPMI, MySQL and RRD databases. It uses the MySQL API and RRD API for interfacing with the databases.

3.1 Detailed Design

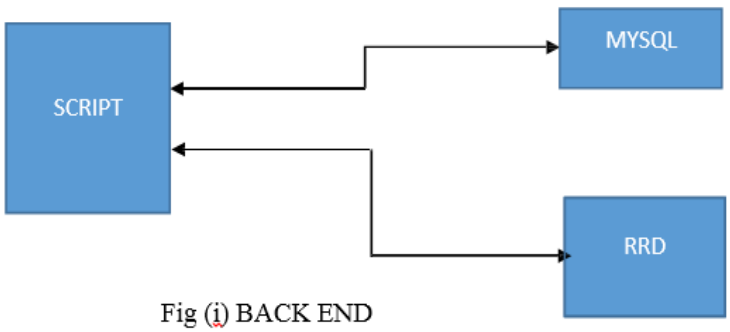

Fig (i) BACK END

This modules takes the HTTP data from the DPMI and performs calculations and statistical analysis. The data stored in the RRD database is used for generating graphs and viewing them in the front end. MySQL database is used to store the login credentials, user details, streams information, server IP addresses and their performance metrics, user defined threshold values and data imported through the RESTful API.

3.2 Unit test plan:

Test	TEST_MOD_1
Purpose	To test the T-shark output
Requirements	USR_REQ_FR1,SYS_FR1,SYS_FR2
Environment	Ubuntu 14.04, LTS
Operation	To Check whether required HTTP packets are filtered or not. In Ubuntu operating system; we capture the packets by using libcap_utils in DPMI. libcap_utils consists of functions such as cap2pcap; cap dump; cap show etc. Capdump generates a .cap file. Here, -i specifies the interface, which is Ethernet 2 of DMPI, from where the TCP packets are being captured. We capture 40,000 packets passing through Port 80, from streams 71 and 72. <capdump -i eth2 –tcp –P 80 –p 40000 01::72 01::71 -o nagios.cap> Cap2pcap converts a .cap file into .pcap. It is done as this file format is accepted by T-Shark. <cap2pcap -o nagiosout.pcap nagios.cap> T-shark is used to filter the HTTP packets. nagiousout.pcap file is given to T-Shark. The command below is used to filter HTTP packets, along with required parameters. It consists of fields like HTTP Request-Response methods; TCP sequence length; TCP acknowledgment number; Epoch time; source and destination IP; source and destination port number; frame length for each HTTP packet all sorted uniquely and store into a text file <tshark -r nagiosout.pcap -T fields -e http.request.method -e http.response.code -e tcp.len -e tcp.seq -e tcp.ack -e frame.time_epoch -e tcp.srcport -e tcp.dstport -e ip.src -e ip.dst -e frame.len | sort | uniq -c > nagiostext1.txt

Expected Result	We successfully filter HTTP packets from captured network traffic.
Result	Successfully, we filtered the HTTP packets using T-shark.
Comment	The user can check the filtering of HTTP packets using Wireshark. Wireshark is a free and open source packet analyzer with GUI, which can be downloaded. Follow the steps below: The user can give the generated .pcap file as input to Wireshark and use the filter HTTP. This gives the details of all the HTTP packets and their corresponding parameters. He can also generate a sample .pcap file using Wireshark and give it to T-Shark, to check if the HTTP packets are filtered or not.

Test	TEST_MOD_2
Purpose	Calculate the performance metrics: 1.Request response time 2.Server bit rate 3.Lost requests
Requirements	USR_REQ_FR2, USR_REQ_FR3, USR_REQ_FR4, SYS_FR2
Environment	Ubuntu 14.04 LTS
Operation	From the log file (nagiostext1.txt), we have written a Perl script to calculate the three above mentioned performance metrics mentioned under “purpose”. The user can run the Perl script “newf.pl”. Request response time: He can subtract the epoch times of the HTTP request with a corresponding response to calculate the request-response time. Server bit rate: The server bit rate is calculated as the ratio of “sum of frame lengths of all the responses sent by a server” to “difference of epoch times of last response and first response. Lost requests: These are the HTTP requests which didn’t receive any HTTP response from the server.

Expected Result	The three performance metrics – Request-Response time, Server bit rate and Lost requests should be obtained.
Result	Request-Response Time, Server Bit Rate, Lost Requests are calculated successfully.
Comment	We tested the functionality of the Perl script "newf.pl" as follows: We generated a sample .pcap file using Wireshark. We noted down the parameters required for our calculations. And manually checked the calculations of Request-Response Time, Server Bit Rate and Lost Requests for 20 IPs and checked the result with the output obtained from the Perl script "newf.pl". This script was then modified to use in real time environment with DPMI. The user can use the sample .pcap file generated in TEST_MOD_1, if possible.

4. MODULE 2: FRONT END

The purpose of this module is to provide the user with a Web Interface to the view the statistical analysis and results. In the high level architecture, this module is present between user and databases. This module interfaces with the MySQL and RRD databases.

4.1 Detailed Design

In the high level analysis, it fetches the data from the database. Its purpose is to serve as an interface between the user and database, so that fetching and accessing of data takes place in the web GUI. We can view the generated graphs and status of the servers.

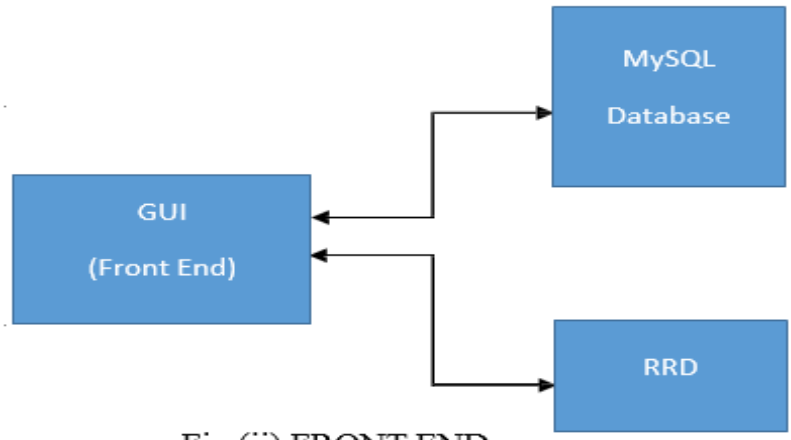

Fig (ii) FRONT END

4.2 Unit test plan:

Test	TEST_MOD_3
Purpose	To test the login authentication
Requirements	USR_REQ_FR8, SYS_FR3
Environment	Ubuntu 14.04 LTS
Operation	The user first registers through the Web GUI. He should enter a proper username and password and click the login button. If the entered username and password match with those on the database, the login is a success. An alert is displayed on the Web dashboard. Else, a failed alert is displayed.
Expected Result	The login should be successful
Result	Login is successful.
Comment	To test the functionality of this module, the user should check whether the details entered by him are successfully being stored in the database. He can access the database through PHPMyAdmin.

Test	TEST_MOD_4
Purpose	To obtain graphs from RRD for the three performance metrics: Request-Response Time, Server Bit Rate, Lost Requests.
Requirements	USR_REQ_FR5, SYS_FR4
Environment	Ubuntu 14.04 LTS
Operation	The user should select the desired server metrics option and the server's IP address. He should be able to see the list of servers and their corresponding IP addresses in the WEB GUI. The graphs must be displayed corresponding to the IP on clicking the submit button.
Expected Result	We must be able to see the graphs on hourly, daily, monthly basis.
Result	Graphs are being displayed successfully.
Comment	To test the functionality of this module, the user should make sure that php5-rrd is installed. The RRDs which are being formed are continuously being updated and the corresponding images are formed. He can use different RRD functions like rrdtool dump, rrdtool fetch to check the values in a particular RRD. (eg). rrdtool dump <rrd-file-name>

Test	TEST_MOD_5
Purpose	To Display MYSQL data in the front end
Requirements	USR_REQ_FR5;SYS_FR1,SYS_FR2
Environment	Ubuntu 14.04 LTS
Operation	When the User accesses the GUI, he should be able to see the IP addresses of all the servers, When he selects the thresholds option, he must be able to see all three performance metrics- Request-Response time, Server Bit Rate, Lost Requests of all IP addresses.
Expected Result	The table containing the Request-Response Times, Server Bit Rate, Lost Requests corresponding to their respective IPs should be displayed in the GUI.
Result	MySQL data is being successfully displayed in the front end.
Comment	To test the functionality of this module, the user should make sure that the database is created, the table containing the performance metric values is not empty. He can access the database through PHPMyAdmin and check this. Also, there should be a successful connection between the front end and MySQL database.

Test	TEST_MOD_6
Purpose	To send a fault notification in the form of an e-mail
Requirements	USR_REQ_FR7,SYS_FR1, SYS_FR2
Environment	Ubuntu 14.04 LTS
Operation	Fault notifications are sent when the threshold limit of the servers reaches a critical state. The user enters his e-mail id during registration. This email id is used as the default email id to send fault notifications. In user-defined threshold values, the fault notifications are sent to the email id provided.
Expected Result	The user must receive an email as soon as the threshold limits are exceeded.
Result	Threshold notifications are being are sent via email.
Comment	

Test	TEST_MOD_7
Purpose	To check whether the threshold levels are set or not
Requirements	USR_REQ_FR6, SYS_FR1, SYS_FR2.
Environment	Ubuntu 14.04,LTS
Operation	If the user wishes to change the default threshold levels, he can do so by assigning in the frontend. The assigned value is compared to the performance metric value in the database. The corresponding threshold level is displayed (using various color codes) in the GUI.
Expected Result	The threshold levels should be displayed as per the specified range (normal-green, critical-orange, warning-red)
Result	The threshold levels are being successfully assigned and displayed in the front end.
Comment	To check the functionality of this module, the user can compare the default threshold values with those being displayed in the table in the front end. Similarly, he can check the user-defined threshold levels.

5. MODULE 3: RESTful API

The purpose of the RESTful API is to interface with the third party user to export and import the data. It is connected to the database at the back end and the third party user.

The RESTful API allows one to retrieve data from our system, at the same time, import data into our system using a URL. Different URL point to different objects. When a URL is retrieved (using the HTTP GET method), data corresponding to that URL is transferred over HTTP connection. If we know the format, then the received data can be converted into a suitable format for storing data in a database or for creating a graph. When data is imported into our system (using the HTTP POST method), data in a suitable file format (eg. .csv file) is transferred into our system through the URL.

We need to export and import data through the API. We implemented the exporting functionality with HTTP GET and importing functionality with HTTP post.

We have documented in our design document and in the developer documentation about how we have mapped URLs to export and import data, and the format of data obtained when accessing the specific URL.

5.1 Detailed design:

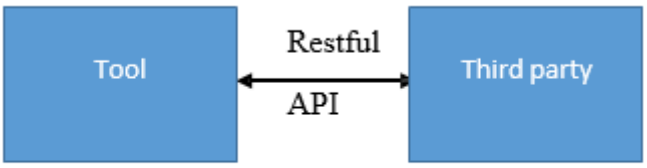

Fig (iii) RESTful API

5.2 Unit test plan:

Test	TEST_MOD_8
Purpose	To export data through a RESTful API
Requirements	USR_REQ_FR9
Environment	Ubuntu 14.04 LTS
Operation	The third party user is provided with a URL to access the desired performance metrics in JSON format: URL: http://"server-IP-address"/web/rest2.php/?value=REQRESP http://"server-IP-address"/web/rest2.php/?value=BITRATE http://"server-IP-address"/web/rest2.php/?value=LOSTREQ This URL connects to the database of the system running the tool, and retrieves the corresponding metrics from the database.
Expected Result	The selected performance metric and the corresponding IP address should be visible to the user in JSON format.
Result	Exporting data to a third party in JSON format via REST API is successful.
Comment	To test the functionality of this module, the user should make sure that a connection is established between the third party user, the system running this tool and the database. He can ping the system. He can compare the values being displayed via REST API with those stored in the database.

Test	TEST_MOD_9
Purpose	To import data through a RESTful API
Requirements	USR_REQ_FR10
Environment	Ubuntu 14.04 LTS
Operation	A third party user can import data into the database of the system which runs this tool. The user uploads a .csv/.json file through a web dashboard, whose contents are stored into the 'CSV_TBL' table in the database. URL: http://"server-IP-address"/web/uploadform.php (for .csv file) http://"server-IP-address"/web/jsonform.php (for .json file) The above URL connects to the database, and transfers the contents of the file into "CSV_TBL" of the database.
Expected Result	The file uploaded by the user should be available in the "web" folder and the contents of the file should be automatically updated in the table "CSV_TBL" of the database.
Result	Importing data in the form of a .csv and .json file via a REST API is successful.
Comment	This API was developed using PHP Curl. It must be installed on the system running this tool. To test the functionality of this module, the user should make sure there is a successful connection between the third party user, the system running the tool and its database.

6. REFERENCES:

[1] Patrik Arlos, Markus Fiedler, and Arne A. Nilsson. *A Distributed Passive Measurement Infrastructure,* In Passive and Active Measurement Workshop (PAM05), US, 2005.

[2] Ian Sommerville, *SOFTWARE ENGINEERING,* 9th ed. Pearson Publications, 2011.