EVALUATING HTTP PERFORMANCE FROM STREAMS

DEVELOPER DOCUMENTATION

- **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 1.2
- **Publication Date:** August 24th, 2015

1. INTRODUCTION:

Developer documentation describes the systematic approach for the developer in the API development. The developer documentation includes source code, data format, Database tables and description of entry tables. This is the second version of the document, v1.2.

The document is partitioned into various sections. Section II describes the glossary and abbreviations. Section III describes the system requirements. Section IV describes the software requirements. Section V describes the system architecture. Section VI, VII, VIII, IX describe the backend, database and front end and RESTful API respectively. Section X describes the organization of our source code. Section XI describes the further extension for our tool. The last section describes the references.

Release v1.2 on 2015-08-24

Updated Release

PUBLICATION DATE	VERSION	DESCRIPTION	CHANGES
2015-08-24	v1.2	Updated version	➢ Organization of source code is included. ➢ Detailed description of RESTful API is included.
2015-06-01	v1.1	Updated version	➢ Architecture of the software is included. ➢ Glossary and Abbreviations are included. ➢ Modules are included. ➢ Description of the RESTful API is included. ➢ References are included.
2015-05-20	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.

RRD: Round Robin Database
This handles the Time series data like data request-response time, Bit rate, standard deviation.

CPAN: Comprehensive Perl Archive Network
This is used for the collection of internet archives.

HTML: Hyper Text Markup Language
It is used to create documents on the web.

CSS: Cascading Style Sheets
It is used for formatting and looking of a document written in HTML.

3. System Requirements:

a) Ubuntu 14.04 LTS version
b) 1 Gb RAM
c) Hard disk 40Gb
d) Processor 32 bit or 64 bit
e) Internet access.

4. Software requirements:

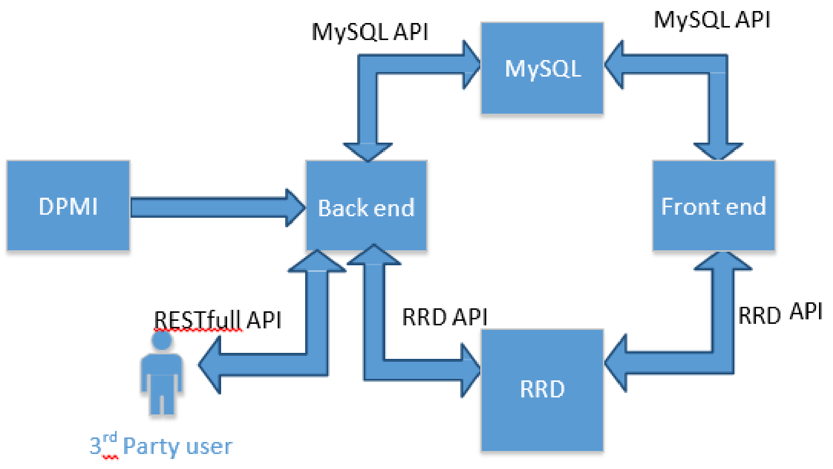
Backend:

1. Perl
2. RRD Simple
3. CPAN: Perl Modules
4. T-shark
5. Libcaputils
6. Apache2 server

Frontend:

1. Web Browser
2. Lamp Server
3. Text Editor (Gedit)
4. Php5
5. RRD
6. MySQL database
7. PhpMyAdmin

5. SYSTEM ARCHITECTURE:



There are four modules in system architecture. They are:-

1. Capture of streaming packets
2. Back end
3. Front end
4. RESTful API

<u>5.1 Module 1:</u>

<u>Capture of streaming packets</u>:

The measurement points (which are a part of the DPMI setup) are placed in between the server and client. The system running our tool is connected to the consumer (which is another part of DPMI). The consumer is connected to the measuring points at one end and to our system at the other end. Thus, we can capture the streaming network traffic through the consumer and measurement points.

5.2 Module 2:
Back end:

In the Backend module, we are provide the basic operation of our tool which is to filter the HTTP packets from the network traffic obtained from DPMI. We calculate the three performance metrics related to a particular server (which is identified based on its IP address). The three performance metrics are: Request-Response Time, Server Bit Rate and Lost Requests. The calculated performance metrics for each server IP address are stored in a MySQL database. Also, in order to provide time series graphs for the performance metrics, it is connected to a RRD database.

5.3 Module 3:
Front end:

In the front end, there is a user authentication which can be used to login into the tool, to view the graphs for the three performance metrics. We can also assign different threshold levels for the performance metrics through the front end. The front end is connected to RRD and MySQL databases.

5.4 Module 4:
RESTful API:

Restful API is used to export and import data to a third party user. A REST API relies on using HTTP methods to manage data.

All the above modules are explained in detail in the following sections.

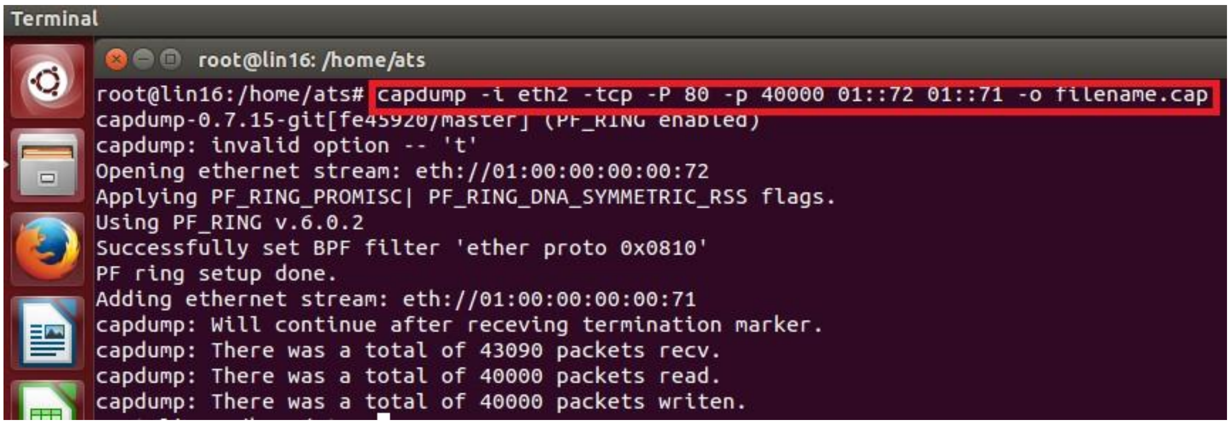
6. Back end:

We consider the DPMI setup which is used to capture the streaming network traffic. A DPMI setup consists of measurement points, Marc and a consumer. Measurement points are used to capture the network traffic. Marc stores the information regarding the interfaces used in the setup. The consumer runs a software called libcaputils, which consists of many functions such as cap2cap, capdump, capshow. The system running our tool is connected to the consumer through an Ethernet cable.

We have captured and filtered the HTTP packets from DPMI using:

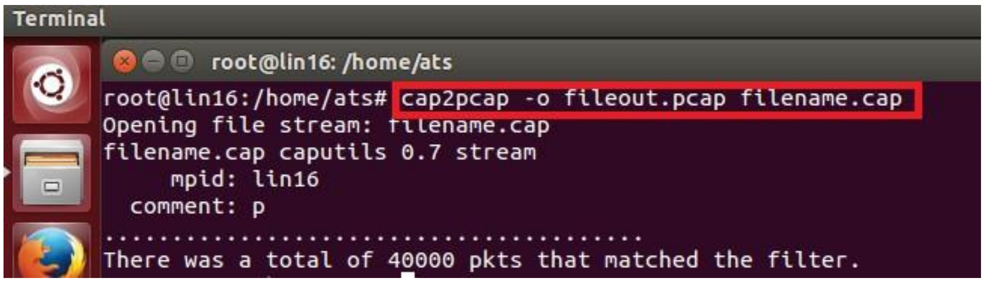
- Capdump
- Cap2pcap
- Tshark

We have used the "**Capdump**" to capture the packets from DPMI and we have stored them in a cap file. For example: **filename.cap**



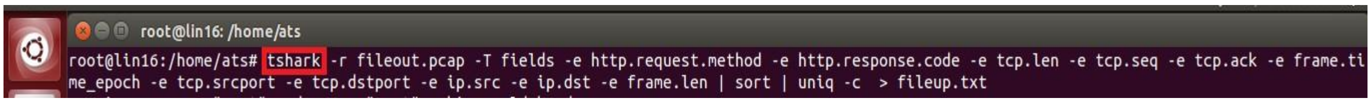
```
root@lin16:/home/ats# capdump -i eth2 -tcp -P 80 -p 40000 01::72 01::71 -o filename.cap
capdump-0.7.15-git[fe45920/master] (PF_RING enabled)
capdump: invalid option -- 't'
Opening ethernet stream: eth://01:00:00:00:00:72
Applying PF_RING_PROMISC| PF_RING_DNA_SYMMETRIC_RSS flags.
Using PF_RING v.6.0.2
Successfully set BPF filter 'ether proto 0x0810'
PF ring setup done.
Adding ethernet stream: eth://01:00:00:00:00:71
capdump: Will continue after receving termination marker.
capdump: There was a total of 43090 packets recv.
capdump: There was a total of 40000 packets read.
capdump: There was a total of 40000 packets writen.
```

The captured packets which are stored in a .cap file are converted to a .pcap file by using the **cap2pcap** command. For example, **fileout.pcap**



```
root@lin16:/home/ats# cap2pcap -o fileout.pcap filename.cap
Opening file stream: filename.cap
filename.cap caputils 0.7 stream
     mpid: lin16
   comment: p
........................................
There was a total of 40000 pkts that matched the filter.
```

The obtained .pcap file has been given to **T-shark** to get the required packets HTTP and its parameters.



```
root@lin16:/home/ats# tshark -r fileout.pcap -T fields -e http.request.method -e http.response.code -e tcp.len -e tcp.seq -e tcp.ack -e frame.ti
me_epoch -e tcp.srcport -e tcp.dstport -e ip.src -e ip.dst -e frame.len | sort | uniq -c  > fileup.txt
```

The output of T-shark is stored in a text file. From the obtained text file, we calculate the required performance metrics using Perl scripting.

Perl scripting is used to calculate the three performance metrics – Request-Response time, Server Bitrate and Lost Requests. These are then used to plot the time series graphs with the help of **RRD tool.** These calculated values are also given to the **MySQL database.**

The Backend Perl Scripts are: “main2.pl”, “newf.pl”, “test.pl”, “mail.pl”, “2.pl”.
All the above scripts are written in Perl.

7. Database:

A database is an organized set of data which can be easily created, modified and deleted. MySQL and RRD are the two database we use in our project.

MySQL database is used to store the values of performance metrics calculated through the Perl script, login credentials, information regarding streams and threshold levels. The database creation, insertion, updation is done automatically through PHP scripts. The database can be accessed through PHPMyAdmin.

The RRD files created in the back end are accessed through PHP scripts to display the time series graphs of the three performance metrics- Request-Response Time, Server Bit Rate, Lost Requests.

7.1 MySQL:

- **Database Configuration:**

There is a database configuration file “db.conf” which is provided along with our tool. The user running this tool needs to modify this file as per his requirement. He needs to change the username and password as per the username and password of his SQL database.
The database “nagios” is automatically created when the user loads the “index.php” webpage for the first time. All the tables are created automatically as well.
The database configuration file “db.conf” is used both for front end and back end scripting.

- **Table Format:**

We have designed six tables. The details are given below:

- ✓ **fish3:** The purpose of this table is to store the information of the registered users. This table consists of columns: id, name, username, password and email. The username and password are verified when a user tries to access this tool.

Table Creation:
"CREATE TABLE if not exists fish3 (id INT(4) NOT NULL PRIMARY KEY AUTO_INCREMENT, name CHAR(30) NOT NULL, usrname CHAR(30) NOT NULL, passwd VARCHAR(50), mail CHAR(30) NOT NULL)"

- ✓ **STREAMS:** This table stores the values of the two streams selected out of the four streams of the DPMI. It consists of three columns, id, STR11, STR2.

Table Creation:

"CREATE TABLE IF NOT EXISTS `STREAMS` (id INT(4) NOT NULL PRIMARY KEY, `STR1` INT NOT NULL,`STR2` INT NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1;"

- ✓ **NEW_IPS:** This table stores the IP addresses of the servers, the three performance metrics: Request-Response Time, Server Bit Rate and Lost Requests. It consists of three columns: DESTIP, REQRESP, BITRATE, LOSTREQ.

Table Creation:

"CREATE TABLE IF NOT EXISTS `NEW_IPS` (`DESTIP` varchar(18) NOT NULL,`REQRESP` float NOT NULL,`BITRATE` float DEFAULT NULL,`LOSTREQ` mediumint(9) DEFAULT NULL, PRIMARY KEY (`DESTIP`), UNIQUE KEY `DESTIP` (`DESTIP`)) ENGINE=InnoDB DEFAULT CHARSET=latin1;"

- ✓ **DEF_THR:** This table stores the default threshold levels for the three performance metrics.It consists of eight columns: id, REQRESP, REQRESP1, SBR, SBR1, LR, LR1, Emailid. The values in this table cannot be changed by the user.

Table Creation:

"CREATE TABLE IF NOT EXISTS `DEF_THR` (id INT(4) NOT NULL PRIMARY KEY,`REQRESP` VARCHAR(225) NOT NULL,`REQRESP1` VARCHAR(225) NOT NULL,`SBR` VARCHAR(225) NOT NULL,`SBR1` VARCHAR(225) NOT NULL,`LR` VARCHAR(225) NOT NULL,`LR1` VARCHAR(225) NOT NULL,`Emailid` VARCHAR(50) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1"

- ✓ **USR_THR:** It consists the same eight columns as DEF_THR but the threshold levels are entered by the user.

Table Creation:

CREATE TABLE IF NOT EXISTS `USR_THR` (id INT(4) NOT NULL PRIMARY KEY,`REQRESP` VARCHAR(225) NOT NULL,`REQRESP1` VARCHAR(225) NOT NULL,`SBR` VARCHAR(225) NOT NULL,`SBR1` VARCHAR(225) NOT NULL,`LR` VARCHAR(225) NOT NULL,`LR1` VARCHAR(225) NOT NULL,`Emailid`

VARCHAR(50) NOT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1"

- ✓ **CSV_TBL:** This table stores the contents of the .csv file imported by a third party user through a REST API.

Table Creation:

CREATE TABLE IF NOT EXISTS `CSV_TBL` (`DESTIP` varchar(18) NOT NULL,`REQRESP` float NOT NULL,`BITRATE` float DEFAULT NULL,`LOSTREQ` mediumint(9) DEFAULT NULL) ENGINE=InnoDB DEFAULT CHARSET=latin1;"

8. Front end:

The front end is a user interface which has to be created in HTML. The required actions are performed by using PHP scripting. The graphs are fetched from the RRD tool. The statistical calculations are provided using RRD only.

The front end provides user authentication to access the required metrics and to provide the thresholds levels. The registration of new user has been provided in this. The user selects the required streams through the front end. The graphs are fetched from the RRD and displayed in the web dashboard. Fault notifications through E-mail are sent when the threshold levels reach the critical stage.

The main parts that are used to present the front end are:

- o HTML
- o PHP
- o CSS
- o JavaScript

The Front End scripts are: “index.php”, “login.php”, index1.php”, “index2.php”, “index3.php”, “graph.php”, “rrtd.php”, “sbrd.php”, “lrd.php”, “unix.php”, “style.css”.

9. 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 of the system running our tool 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 URLs. Different URLs point to different objects.

- **Exporting Data:**

This functionality is implemented using HTTP GET Method. The third party user is provided with three URLs, each URL corresponding to a particular performance metric. We have three performance metrics – Request-Response Time, Server Bit Rate and Lost Requests:

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

When the 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. Here, we display the output in JSON format.

JSON - JavaScript Object Notation, is an open standard format that uses human-readable text to transmit data objects consisting of attribute–value pairs. It is used primarily to transmit data between a server and web application. It is a language-independent data format. The official Internet media type for JSON is `application/json`. The filename extension is `.json`.

Sample JSON format:

```
{"status":200,"status_message":"REQRESP metrics Found"}
"109.105.109.10"                    "0.417183"
"109.105.109.18"                    "0.017803"
```

The corresponding scripts are: “rest2.php”, “database2.php”, “details.php”

- **Importing Data:**

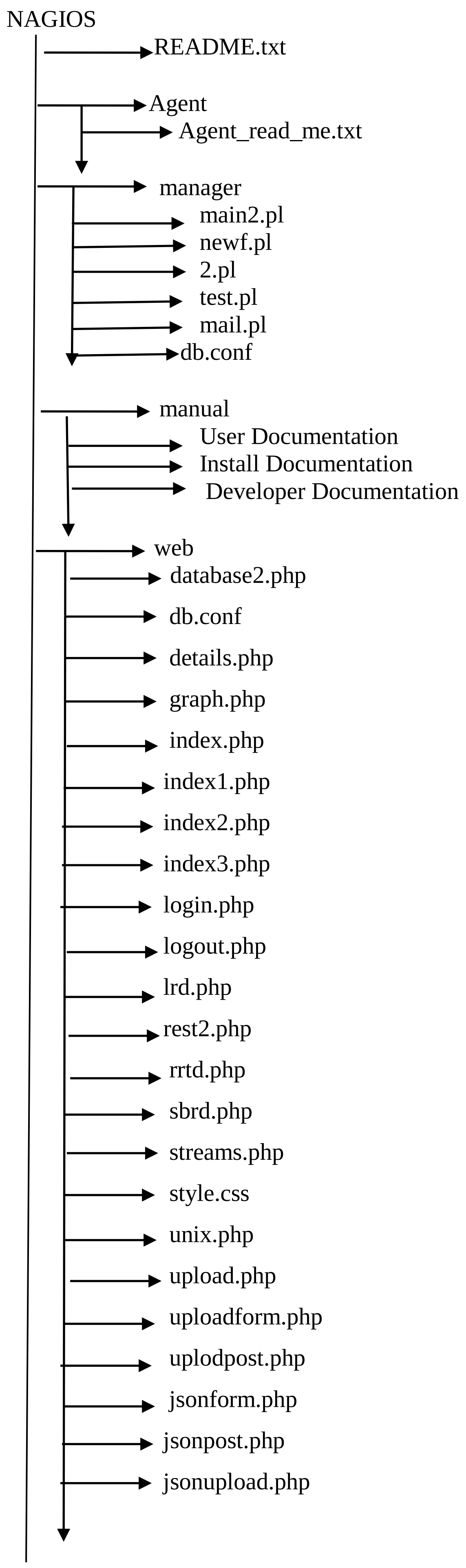
The data is imported into our system (using the HTTP POST method) using a different URL. Data in a suitable file format (eg. .csv or .json file) is uploaded into our system and the contents of the file are stored in SQL database. This functionality is implemented using PHP Curl.

The user is simply provided with a URL to a webpage where he can upload the file. The file data is carried through HTTP POST into our system.

URL: http://”server-IP-address”/web/uploadform.php (for importing .csv file)
http://”server-IP-address”/web/jsonform.php (for importing .json file)

The corresponding scripts are:” upload.php”, “uploadpost.php”, “uploadform.php”
“jsonupload.php”, “jsonpost.php”, “jsonform.php”

10. Organization of Source Code



11. Extension for the Tool

This tool can further be extended. It is possible to include receiving of fault notifications through SMS service. At the same time, a better method such as introducing a buffer to store the output of T-shark can be used, instead of storing the output in a log file. The feature of importing through a REST API can be further be improved by introducing the ability to import other file formats such as .xml.

12.References:

[1] http://php.net/manual/en/book.rrd.php

[2] DPMI: Distributed Passive Measurement Infrastructure,
URL: https://github.com/DPMI/libcap_utils