EVALUATING HTTP PERFORMANCE FROM STREAMS

PROJECT PROPOSAL

- **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:** Project Proposal
- **Version Number:** Version 1.4
- **Publication Date:** May 14th, 2015

1. INTRODUCTION

In this document, we present a proposal on how to evaluate HTTP performance from streams. From the feedback of CEO, the updated version release is v1.4 where we elucidate on how to develop a tool to monitor the HTTP traffic.

The document is partitioned into various sections. Section I gives an overview of the customer's needs and the urge to develop a new product. Section II suggests a solution addressing the demands of the customer. The limitations of what is included and excluded in this project are explained in Section III. Section IV depicts the work breakdown structure describing the toll gates and milestones.

Release v1.4 on 2015-05-14

- Updated release

Version history is as follows:

PUBLICATION DATE	VERSION	DESCRIPTION	CHANGES
2015-05-14	v1.4	Updated version	Updated the following from feedback of the CEO. • Changes made to HTTP request under system architecture. • RESTful API under proposed solution.
2015-05-05	v1.3	Updated version	Updated the following from feedback of the CEO. • Changes made to the system architecture. • RESTful API
2015-04-27	v1.2	Updated version	Updated the following from feedback of the CEO. • Measuring the performance of the servers.

			• The data rate being delivered to the user is not affected. • Changes in tollgates and milestones.
2015-04-20	v1.1	Updated version	Updated the following from feedback of the CEO • Fault Notifications • RESTful API • Performance and aggregation of servers. • GUI
2015-04-13	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.

3. BACKGROUND

The customer is a data centre company providing rack space and rudimentary networking to clients. They are basically web service providers. The increased demand in live streaming over HTTP generates more traffic towards the servers.

Thus, the clients have expressed their displeasure over the services provided to them. Slow request-response time, poor service quality are a few of such problems. There is an overall reduction in the quality of the internet services provided to the end users.

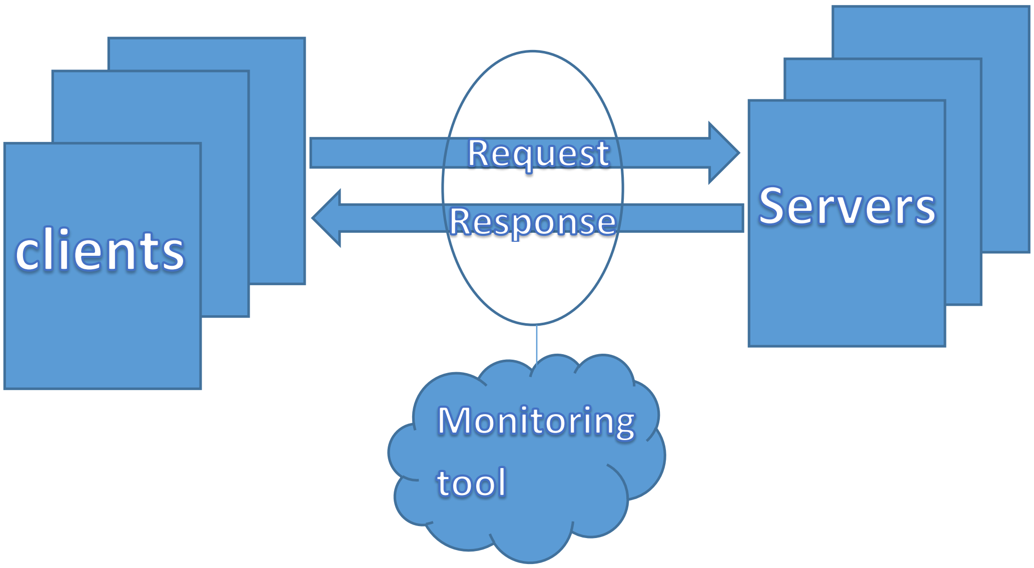
The customer requires a solution to the above issues along with a picturesque way to exhibit these improved services with the help of a tool. Quick responses are expected. This tool thus monitors the HTTP performance of the servers present in the data centre.

4. PROPOSED SOLUTION

The following methods are devised to meet the requirements of the customer.

System architecture

A typical internet consists of multiple clients and servers. The clients are users. The servers are located at the data centre.



In the above figure, an HTTP request is **for** an HTML file, a document or a movie**.** An HTTP response carries the data back (which is requested) or an error (when the resource is not available). In some cases, the response carries a redirection code, when the resource is moved.

We plan to develop a monitoring tool which observes the HTTP request-response time per server, bit rate and lost requests. This tool maintains a historical track record for about four weeks. We use a DPMI to read the data stream. We use time series graphs to represent this data. Statistical inference like standard deviation and confidence analysis can be carried out on this data for further investigation.

The performance of the servers is observed. We calculate the request-response rate, bit rate and lost requests associated with each server. We plot a graph which displays these performance metrics of all the servers in the data centre. We also display these metrics for a cluster of servers for comparison. A cluster of servers is a group of independent servers working together to provide high availability of services to the clients.

We assign threshold levels to the servers. The threshold levels are normal, warning and critical. A notification is sent when threshold limits are exceeded. These threshold limits are configured for each device which we are monitoring.

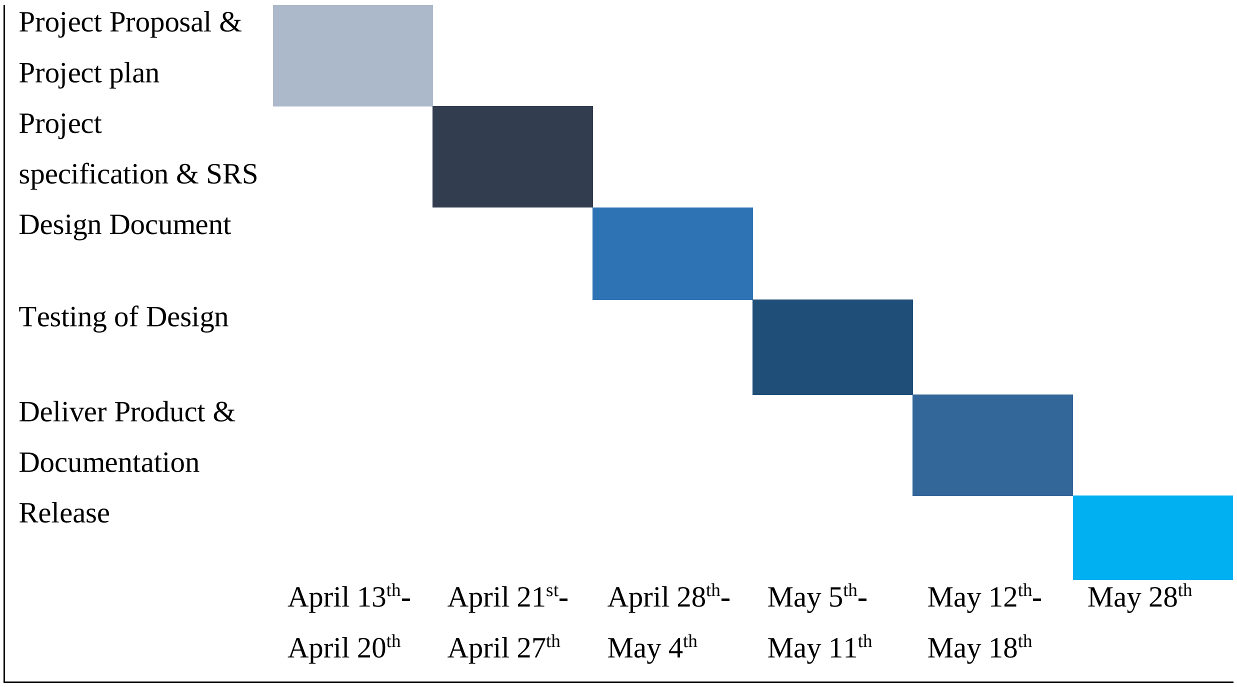
A simple web interface with user authentication will be provided to display the observed statistics in the form of graphs. After successful login, a dashboard is provided which displays the critical problems associated with each server. We can also drill down and zoom-in on to each device individually for detailed information.

RESTful API is an interface which can be used to export data to a third party and import data from the third party in any agreed-upon format (for example, time series data or any other format). Graphical cross correlation between our data and provided data can be performed for further analysis.

5. LIMITATIONS

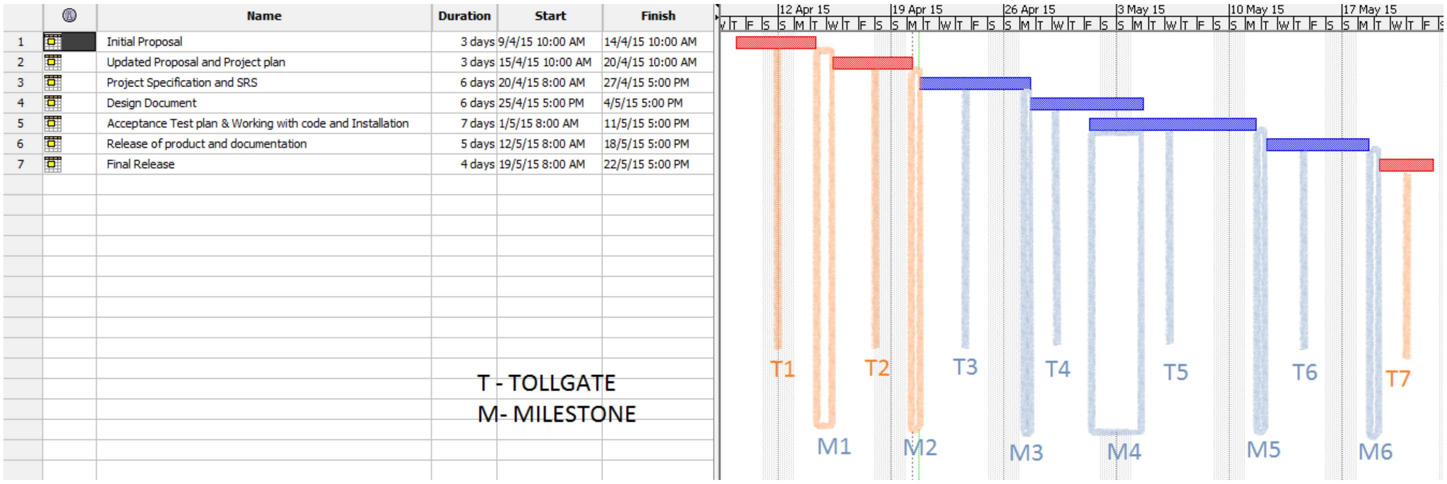
The tool is for monitoring the HTTP traffic over multiple data streams. It is applicable for monitoring of services like request-response time. It informs the faults to the observer by issuing proper fault notifications. The fault notifications can be sorted by activating a software that allows us to take proper intervention i.e., send E-mail. This tool cannot capture the entire user information. Only truncated packets are available for observation. Sampling of raw data obtained at various measuring points might be required. Modelling the client behaviour is also an important limitation. Server performance metrics like disk usage and memory utilisation and also network metrics like delay, reliability and error rate are beyond the scope of this project.

6. TIME PLAN



A Gantt chart showing the proposed time plan for project completion

- 13-04-2015: Project Proposal
- 20-04-2015: Project Plan
- 21-04-2015: Project Specification
- 27-04-2015: Software Requirement Specifications (SRS)
- 04-05-2015: Design Document
- 11-05-2015: Testing of Design, Project Demo
- 18-05-2015: Acceptance of design plan, deliver product, documentation
- 28-05-2015: Release



A work break down structure is designed to meet the deadlines within the allotted time. The various targets to be achieved are classified into milestones (for the development team) and milestones (for the customer/CEO).

The project duration is of six weeks. Each developer is supposed to work for 8hours/business day. This is equivalent to 30days/240 hours per person. The entire team is supposed to work for sixty weeks on a full time basis (as on average, the members per team are 10). This time is equivalent to a one and a half man year.

The below table gives the division of Milestones and Tollgates:

Milestones:

Milestone 1	Project Plan
Milestone 2	Design Document
Milestone 3	Project Documentation

Tollgates:

Tollgate 1	Project Proposal
Tollgate 2	Software Requirement Specification
Tollgate 3	Acceptance of Design Plan
Tollgate 4	Final release of the Product

Table: Milestones and Tollgates

Milestones are primarily concerned with the project team. But, when it comes to project plan, both the CEO and Customer are involved. Similarly, in Tollgates, the acceptance of design plan involves discussion between the team members, Customer and CEO. The final product needs to be accepted both by the customer and CEO.