

EPM - End-to-end Performance Measurement

By

Fahad Ahmed Satti

2004-NUST-BIT-234



Project Report in partial fulfillment of the requirements for the award of
Bachelor of Information Technology degree

School of Electrical Engineering and Computer Science
National University of Sciences and Technology
Rawalpindi, Pakistan
2008

EPM – End-to-end Performance Measurement

By

Fahad Ahmed Satti

(2004-NUST-BIT-234)



Project Report in partial fulfillment of

the requirements for the award of

Bachelor of Science degree in Information Technology (BIT)

In

School of Electrical Engineering & Computer Science (SEECS)

National University of Sciences and Technology (NUST)

Rawalpindi, Pakistan

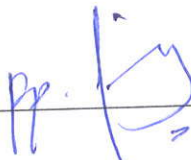
(2008)

CERTIFICATE


It is certified that the contents and form of thesis entitled “**EPM – End-to-end Performance Measurement**” submitted by Fahad Ahmed Satti have been found satisfactory for the requirement of the degree.

Advisor:  _____

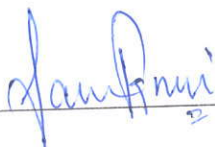
Faculty Member (Mr. Ali Sajjad)

Co-Advisor:  _____

Faculty Member (Mr. Umar Kalim)

Committee Member:  _____

Faculty Member (Dr. Aamir Shafi)

Committee Member:  _____

Faculty Member (Mr. Savera Tanwir)

DEDICATION

In the name of Allah, the Most Gracious, the Most Merciful.

To my dearest

Father, mother and brothers!!

ACKNOWLEDGEMENTS

Firstly I am thankful to ALLAH Almighty, for bringing the project to the current state. I am also thankful to my mom for staying up and letting me in the house even though no one knew when I returned back. I am thankful to my dad for supporting me and continuously asking me questions and guiding me. Without him I would not even be here in the first place. I am also thankful to my siblings for not bugging me when I was working, comrade z for actually keeping up with me, and enduring all that I threw at her.

I am thankful to my project supervisor Sir Umar Kalim, Dr. Less Cotrell, Yee, Jared, Sir Muhammad Atif, Sir Atif Kamal, Sir Aamir, Sir Ali, Sir Shahrzad Khattak, Sir Muaz Niazi, Miss Sana Khalique, Miss Shumaila, Sir Saad Liaquat, Sir Nauman Qureshi, Dr. Arshad Ali and over hundred other teachers all over Pakistan who made me what I am today.

Also, I would like to pay special thanks to Shahry Bhai, Asif bhai, Khawar bhai, Mojo bhai, Alvi bhai, Chauhdry bhai, Latif bhai and all those seniors whose I name I don't remember now. I am thankful to my class, particularly Badar who helped me start up, lan, saji, sufi, jd, shah, doc, meesam, sheri plz, aqcel, asma(minimi), sam, fauzia(nurse) and everyone. All my class mates were really helpful and real good friends. I am thankful to all the juniors who taught me a lot; I hope I did the same. And finally I am thankful to Dr. Arshad Ali for providing me with so many opportunities, and for being so supportive.

I just hope I haven't missed anyone, if I have then I am sorry

TABLE OF CONTENTS

TABLE OF FIGURES.....	5
INTRODUCTION.....	6
1.1 PROBLEM STATEMENT.....	6
1.2 INTRODUCTION.....	6
LITERATURE REVIEW.....	8
2.1 BACKGROUND.....	8
2.1.1 IPMA.....	10
2.1.2 AMP.....	10
DESIGN.....	11
3.1 SYSTEM DESIGN.....	11
3.1.1 Level 0 or Monitored Node.....	12
3.1.2 Level 1 or Monitoring Host.....	12
3.1.3 Level 2 or Archive Server.....	13
3.1.4 Level 3 or User.....	13
3.2 THE DATABASE.....	14
3.2.1 "Data" Table.....	15
3.2.2 "Test" Table.....	16
3.2.3 "TestGroup" Table.....	17
3.2.4 "Tool" Table.....	17
3.2.5 "Metric" Table.....	17
3.2.6 "Path" Table.....	17
3.2.7 "Node" Table.....	17
3.2.8 "NodeDesc" Table.....	18
3.2.9 "monhosts" Table.....	18
3.2.10 Summary Tables.....	18
3.2.10.1 Summary tables on monitoring hosts.....	19
3.2.10.1.1 Raw summary tables.....	19
3.2.10.1.2 Summary tables with hourly aggregated values.....	19
3.2.10.2 Summary tables on archive servers.....	20
3.3 SCHEDULING.....	21
IMPLEMENTATION.....	23
4.1 TESTER.....	23
4.2 DATA ARCHIVAL.....	24
4.3 THE WEB INTERFACE.....	25
4.4.1 MonHost.....	31
4.4.2 Archive Server.....	32
FUTURE WORK AND CONCLUSION.....	33
5.1 CONCLUSION.....	33
5.2 FUTURE WORK.....	33
REFERENCES.....	35

TABLE OF FIGURES

Figure 1: EPM Abstract Architecture	11
Figure 2: MonIlost Database	15
Figure 3: monhosts Table on Archive Server	18
Figure 4: Summary Table Format.....	19
Figure 5: epm-mon.config file format	23
Figure 6: epm-arc.config file format.....	25
Figure 7: Main web interface	26
Figure 8: web interface- MonIlost selected	26
Figure 9: web interface-summary table selected	27
Figure 10: web interface-timestamp in human readable form	27
Figure 11: web interface-timestamp selected	28
Figure 12: web interface-MonIlost selected	28
Figure 13: web interface-search criteria complete.....	29
Figure 14: web interface-normal results	29
Figure 15: web interface-results with timestamp in formatted form	30

INTRODUCTION

This document contains design and implementation details for the project EPM. The document is divided in Chapters, which provide, Introduction, Literature survey, Design and Implementation details.

This Chapter explains the problem at hand and introduction to network performance measurement.

1.1 PROBLEM STATEMENT

The performance of the network depends on its bandwidth or throughput, as well as factors such as network delay, loss of data packets, and network jitter etc.

Independent tools exist which gather such metrics, but Network Analysts require a single platform that can provide the same information in an automated manner.

EPM project aims to develop an end to end Computer Network Performance Measurement infrastructure by employing sophisticated performance measurement tools. The infrastructure may later be enhanced by incorporating analysis algorithms.

EPM intends to provide a single platform that can accommodate most of the tools and network performance metrics. The objectives are two-fold a) collection of data and b) presentation of records. Automated monitoring processes will ensure that minimum management is required by the users. This way the Network analysts can focus on using the data rather than collecting it.

1.2 INTRODUCTION

Enumerating the network performance and evaluating its behavior is a non-trivial task. Each Computer Network has some parameters associated with it. These include available

Bandwidth, throughput and jitter. The most trivial active measurement tools such as PING, Traceroute and Pathload send data to peer nodes and analyze the response and associated parameters to estimate performance metrics. Executing such tools and extracting results over long durations, so as to analyze the network, is a cumbersome job. This job becomes almost impossible if more than one tool is used to calculate a single metric. Thus there is a need of software that can undertake this laborious task for the network analyst i.e. conducting the tests, gather responses and maintain the context in which the results were gathered.

Over the years scientists and computer programmers have devised such software (some of these will be discussed later), that manage the repeated steps in order to extract the required metrics.

This said, the need is to develop generic software that can integrate most of the tools and extract useful information. Here information comprises of the network performance metrics that a network analyst wants to see over some period of time thus allowing the Analyst to observe trends which may help in planning, capacity building as well as maintaining operations.

Tools such as PingER[1], IEPM[2] and PerfSONAR[3] have been collecting statistics reflecting network metrics for quite some time now, however each is geared towards an independent aspect of network performance measurement. We focus on IEPM-like infrastructures.

LITERATURE REVIEW

EPM takes its inspiration from several projects, primarily from IEPM-BW. These are discussed in detail next.

2.1 BACKGROUND

IEPM-SLAC has established a large infrastructure for network performance measurement all over the world. This infrastructure provides near real time network performance related data. IEPM is also developing analysis techniques and tools that detect drop in performance. Being part of the MAGGIE initiative (the collaboration between NIFT and SLAC IEPM) this project will spearhead the development of these algorithms.

EPM takes its inspiration from IEPM-BW (Internet End-to-end Performance Monitoring - Bandwidth to the World) developed by SLAC. The purpose of the IEPM-BW project is to develop and use an infrastructure to make active end-to-end application and network performance measurements for high performance network links such as are used worldwide by Grid applications and other academic and research (A&R) applications deployed over high performance network such as ESnet, Internet2 and other (A&R) networks in the developed world.

Some of the tasks defined by IEPM-BW project were:

1. Develop/deploy a simple, robust, ssh based active end-to-end application and network measurement and management infrastructure.
2. Install/integrate a base set of measurement tools into the infrastructure make regular measurements and record the results. These tools include: ping, traceroute iperf, piperchar, bbep and bbfp.
3. Develop data reduction, analysis, reporting, forecasting and archiving tools.
4. Compare and validate the various tools and determine the regions of applicability.
5. Install new network (e.g. the INCTE tools, pathrate and pathload) and application (e.g. GridFTP) tools into the infrastructure, and use it to evaluate the performance of the tools and their relevancy.

6. Evaluate new TCP stacks such as HTCP, High Speed TCP, and FAST and compare with the default stacks.
7. Provide access to the data for research, forecasting, validation etc

IPEM-BW was developed incrementally to achieve its objectives which it does amiably. however this incremental development has resulted in a system which is nearly impossible to manage and/or extend. The System consists of numerous scripts, reports, directories and a relational database that is hard to understand and extend. This makes the system too complicated for deployment, even a good administrator would require some time with the system to understand the intricacies of its deployment. One of the main hindrances is the non availability of a complete deployment document. At first it was thought that maybe just putting a new layer of scripts over the existing ones would solve the problem. The idea was to develop and deploy some easier to understand scripts using already existing scripts in addition to a layer of tables over the existing database. These new scripts would help achieve some sort of flexibility and robustness but the system as a whole would still remain difficult to deploy and hard to extend.

The most feasible solution was to redesign the system from scratch, using functionalities of higher level languages and new DBMS.

In addition to the tasks defined by IPEM-BW, some new features, in the new system, were formulated. These are:

1. The system should be able to generically integrate Network Monitoring and Performance Measurement Tools, like PING, Traceroute, Pathload, Pathchirp, Thrulay TCP and iperf.
2. The system should be easy to extend.
3. Data should be extracted using methods defined by user.
4. Data should be reliable, available through multiple means and in a useable format. Data context to be maintained at all times, while keeping resource consumption to a minimum level.
5. The system should not be resource hungry.
6. The system should be easy to deploy.
7. The system should be automated, thus requiring minimum human interaction.

6. Evaluate new TCP stacks such as HTCP, High Speed TCP, and FAST and compare with the default stacks.
7. Provide access to the data for research, forecasting, validation etc

HEPM-BW was developed incrementally to achieve its objectives which it does amiably, however this incremental development has resulted in a system which is nearly impossible to manage and/or extend. The System consists of numerous scripts, reports, directories and a relational database that is hard to understand and extend. This makes the system too complicated for deployment, even a good administrator would require some time with the system to understand the intricacies of its deployment. One of the main hindrances is the non availability of a complete deployment document. At first it was thought that maybe just putting a new layer of scripts over the existing ones would solve the problem. The idea was to develop and deploy some easier to understand scripts using already existing scripts in addition to a layer of tables over the existing database. These new scripts would help achieve some sort of flexibility and robustness but the system as a whole would still remain difficult to deploy and hard to extend.

The most feasible solution was to redesign the system from scratch, using functionalities of higher level languages and new DBMS.

In addition to the tasks defined by HEPM-BW, some new features, in the new system, were formulated. These are:

1. The system should be able to generically integrate Network Monitoring and Performance Measurement Tools, like PING, Traceroute, Pathload, Pathchirp, Thrulay TCP and iperf.
2. The system should be easy to extend.
3. Data should be extracted using methods defined by user.
4. Data should be reliable, available through multiple means and in a useable format. Data context to be maintained at all times, while keeping resource consumption to a minimum level.
5. The system should not be resource hungry.
6. The system should be easy to deploy.
7. The system should be automated, thus requiring minimum human interaction.

Some other Network Performance Measurement projects are discussed next:

2.1.1 IPMA

The pioneering Internet Performance Measurement and Analysis (IPMA) project, a joint effort of the U-M Department of Electrical Engineering and Computer Science and Merit Network, helped lay the foundation for data collection and statistical analysis in the Internet. The three-year project was launched by a \$1.6 million award from the National Science Foundation, following NSF's recommendation that Merit pursue statistical research and tool development separately from the Routing Arbiter activity.

The IPMA project focused on two primary areas of Internet statistics: routing stability, topology, and visualization; and ISP performance measurements. The overall goal of the project was to develop tools and perform statistical research that promotes the stability and rational growth of the Internet[4].

IPMA however developed its own Tool set and was not designed to conduct end to end tests. IPMA focused on broader audience, by monitoring network performance over the internet.

2.1.2 AMP

The Active Measurement Project (AMP) was designed and implemented with a joint research/engineering focus in mind. AMP provided site-to-site active measurements and analyses conducted between campuses connected by high performance networks.

AMP architecture consisted of meshes and was more concerned with monitoring links between domains. Measuring RTT, packet loss, topology and throughput (user/event driven). AMP provided data repositories for easy and fast access. AMP has however ceased its active monitoring, due to resource constraints. AMP also proposed utilizing a new protocol IPMP (IP Measurement Protocol), to overcome some of the problems with ICMP[5].

DESIGN

This chapter discusses the architectural details of EPM. It is divided in three parts; the first part discusses System Design, followed by database design and finally the scheduling methodology.

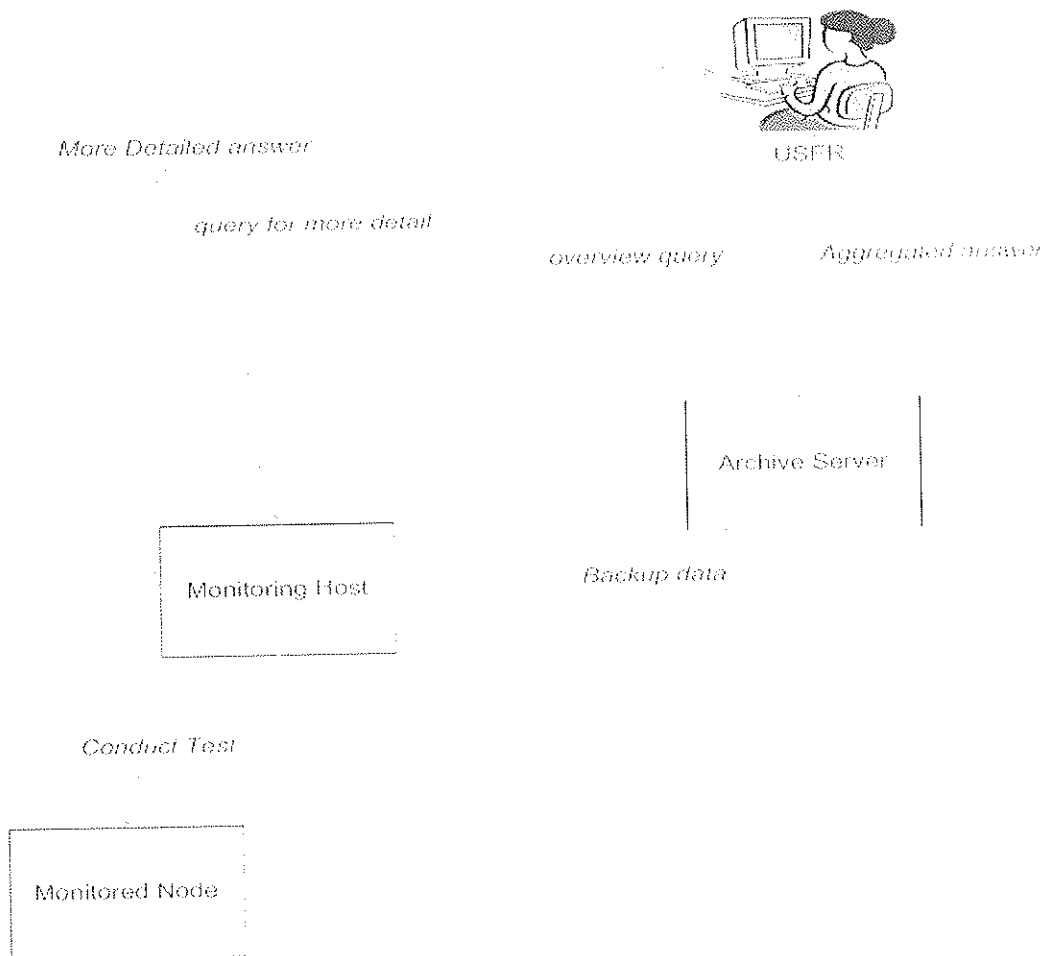


Figure 1: EPM Abstract Architecture

3.1 SYSTEM DESIGN

EPM comprises of four levels and three paths between these levels. These are:

3.1.1 Level 0 or Monitored Node

Level 0 or Monitored Node is the client system which acts as the target machine for any network related test. This machine can be any router, server or a client machine that has the capability to answer a request generated by some higher level System.

The Monitored Node is equipped with client side part of the Tool that is being used by the Monitoring Host to conduct some test. This means if the Monitoring Host is just sending PING request to the Monitored Node then the Monitored Node must have ICMP traffic enabled. Monitored Node can be any client machine it just has to satisfy the needs of the Tools being used.

3.1.2 Level 1 or Monitoring Host

The Monitoring Host is a node that conducts the tests, extracts information and then shares it. A Monitored Host requires installation of some applications, a database and web interface.

A "Path" is defined as the connection between the Monitoring Host and the Monitored Node. A Monitoring Host conducts its tests on a path; where the Monitoring Host initiates a test and the monitored node replies to it.

The term "End to end" implies that the monitoring host wants to test the path; irrespective of the distance of the path. So when we define a path we actually define the two ends of a path and nothing in between.

In case of PING the Monitoring Host must have the script to conduct ping test in addition to other required interfaces.

It requires as mentioned above the server processes of the tools employed, a database, that contains information useful for EPM, summary tables, applications to execute tests, applications that manage the database and its entries, and a web interface.

Every Monitoring Host contains a database that has some summary tables which contain the raw metrics, i.e. all metrics extracted from the tests, and also hourly aggregation of these metrics in separate tables. But a Monitoring Host contains only the metrics it collected itself.

3.1.3 Level 2 or Archive Server

Level 2 or archive server is an optional but very important part of EPM. EPM has the capability to operate without an archive server; using just the Monitoring Hosts and the Monitored Node. However this design won't allow sharing of data with other networks as well as between the Monitoring Hosts. Hence although possible it is suggested that, in order to make sure good usage of data; archive server should be included.

The Archive server acts as the central storage for all the monitoring hosts falling under its domain. This means that every enterprise can have just one archive server, which it uses as the central storage. The archive server not only increases the availability by putting redundancy but also allows for an external interface for users to view the information.

It is expected that after the whole system is in place the users will have a public view of all the data placed in the archive server. This data will be used primarily however to get more details the user will need to connect to the Monitoring Host to collect data.

The Archive server consists of a database containing raw metrics from all the Monitoring Hosts falling under its domain, in addition to some summary tables again for all the Monitoring Hosts falling under an Archive Server's domain. Some scripts are in place to manage the database and populate the Tables.

The Archive server will also contain the external interface for the information in the database. It is expected that the interface will be a website hosted on the archive server, providing access to the information.

3.1.4 Level 3 or User

The user is the Network Analyst or Administrator who is interested in the Network Metrics; he will be able to view the information by connecting to the Archive Server or the Monitoring Hosts depending on the amount of details he requires.

Every Monitoring Host contains a database that has some summary tables which contain the raw metrics, i.e. all metrics extracted from the tests, and also hourly aggregation of these metrics in separate tables. But a Monitoring Host contains only the metrics it collected itself.

3.1.3 Level 2 or Archive Server

Level 2 or archive server is an optional but very important part of EPM. EPM has the capability to operate without an archive server: using just the Monitoring Hosts and the Monitored Node. However this design won't allow sharing of data with other networks as well as between the Monitoring Hosts. Hence although possible it is suggested that, in order to make sure good usage of data; archive server should be included.

The Archive server acts as the central storage for all the monitoring hosts falling under its domain. This means that every enterprise can have just one archive server, which it uses as the central storage. The archive server not only increases the availability by putting redundancy but also allows for an external interface for users to view the information.

It is expected that after the whole system is in place the users will have a public view of all the data placed in the archive server. This data will be used primarily however to get more details the user will need to connect to the Monitoring Host to collect data.

The Archive server consists of a database containing raw metrics from all the Monitoring Hosts falling under its domain, in addition to some summary tables again for all the Monitoring Hosts falling under an Archive Server's domain. Some scripts are in place to manage the database and populate the Tables.

The Archive server will also contain the external interface for the information in the database. It is expected that the interface will be a website hosted on the archive server, providing access to the information.

3.1.4 Level 3 or User

The user is the Network Analyst or Administrator who is interested in the Network Metrics; he will be able to view the information by connecting to the Archive Server or the Monitoring Hosts depending on the amount of details he requires.

3.2 THE DATABASE

The Database can be divided into two parts, raw data table and summary tables. The Raw data tables provide data necessary for conducting tests. This data for example is the location of the tool to be executed, the parameters it requires for execution, the user defined code for extraction of metrics and much other related data. The second part consists of summary tables. These tables contain results, or the extracted network metrics and the primary attributes associated with each test.

The Database can also be categorized in terms of the design level it falls in i.e. the tables on the MonHosts and the tables on Archive Server. The tables on the MonHosts are used primarily to conduct tests and provide a primary storage area for network metrics. The tables on Archive Server, on the other hand, are used as a central repository for users to access network metrics.

Description of each table follows:

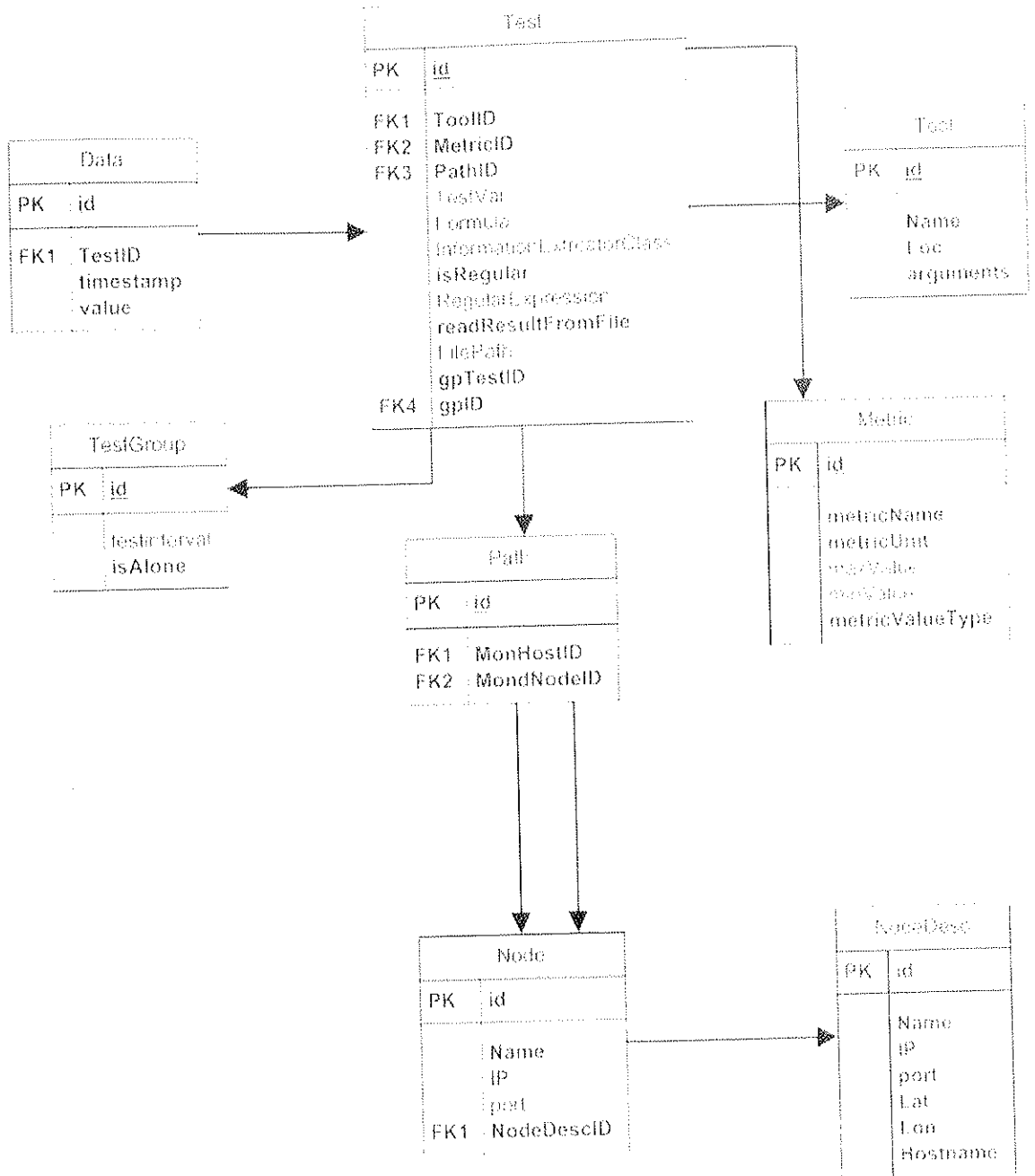


Figure 2: MonHost Database

3.2.1 "Data" Table

The most prominent part of our database is the fact that the network metrics are stored generically in one table and are not stored separately against the Tool that was used or the metric name. This saves the time of searching the table name before making insertions. A particular

metric value in a data table is made unique by the id field; however the metric value is associated by the test that was used to find out the metric and a timestamp at which the test was performed.

The Data table does round robin storage and is flushed after 1 day. So the size of the data table is kept small and in control.

3.2.2 "Test" Table

The Test table contains the test entries. A test is defined by a Tool used to extract a Metric on a Path. This means RTT (Metric) using Ping (Tool) between xxx.xxx.xxx.xxx and yyy.yyy.yyy.yyy (Path) is a Test table entry. This table contains information on how a metric and Tool are related and how to extract information from data received as result of a Test conducted by a Tool. In order to extract a metric from the result there are two ways, one of them is using the regular expression on the result string (This method just searches for a given string in the result string). The other method is using the user defined methodology of extracting metric from the result. The result string is the output of the Test conducted with line breaks replaced by spaces. The output can be on the screen or in a file; "readResultFromFile" defines which methodology to use and "FilePath" defines the path where the file can be found. The user provides the methodology of extracting metrics by implementing the abstract class "AbstractDataExtractor". This class consists of an abstract method "processdata"; this method then contains the actual methodology; it receives a string containing the result of the Test and returns the metric as string value to be stored in the Data table. The name of this class is placed in the field "Information\extractorClass". Note that the name must be placed in particular format that enables java to dynamically load the class file. This format is "packagename.classname". EPM contains a package by the name of "Metric\extractorUtil" this package is converted to jar and deployed with the system. This package contains the abstract class as well as the other implemented classes for extracting the metric. The type of methodology to use; in order to extract the metric; is defined by "isRegular" field. The value of this field is 1 in case regular expression are used otherwise if Extractor class is to be used the value is 0. "gpID" defines the group this Test belongs to, while "gpTestID" defines the execution sequence of Tests, The Test with the lowest gpTestID is executed first and then so on.

3.2.3 "TestGroup" Table

The TestGroup table consists of data about the whole Group. Each Group is currently executed in a separate thread, "testinterval" is used to define the time when this test should run (but has not yet been implemented). The variable "isAlone" defines if this test is to be run independent of the other tests or not.

3.2.4 "Tool" Table

The Tool Table contains description of a particular tool; like the dir it is in and the arguments it is to be run with; e.g. in case of Ping the tool table will have its name as "ping" its location can be "/bin/ping" and the arguments can be "-c 5". However the arguments must contain a string "clientip" separated by spaces. In the application used to conduct test this string is replaced by the monitored node's IP.

3.2.5 "Metric" Table

The Metric table consists of metadata related to metrics. These include name of the metric, its minimum value, its maximum value, its unit and its data type. The "metricValueType" field is an enumeration which can contain any one of the "double", "string" or "integer" type. In case of "rtt" we can have the name as "rtt" or "round trip time", its minimum value can be 0 and maximum 1000, its unit sec or msec and its data type is double. This provides all the information and more, about the metric "rtt" that is required by EPM.

3.2.6 "Path" Table

The path table consists of the Monitoring Host's and the Monitored Nodes References. What this means is that a path is a logical connection between two end nodes, a MonHost and a MondNode. This path is just the logical representation, and not necessarily a physical one. Path represents a connection on application layer between the MonHost and the MondNode.

3.2.7 "Node" Table

The Node Table contains general information about a node; whether it's monitoring host or a monitored node. This information is used for testing. It contains the name of the host, its IP

address and the port with external interface. It also references a NodeDesc Table entry which contains a bit of detailed information about the node.

3.2.8 "NodeDesc" Table

This table contains some detailed information about the node; which includes its latitude, longitude and hostname along with Node Table entries. This table will consist of more details in future; that may be required by the user.

3.2.9 "monhosts" Table

This table is present on the archive server, this table consists of entries of all the MonHosts that exists under an archive server and that can share their data with it.

This table is used by the Archive server to keep a check on the MonHosts so as to authenticate any data transfer occurring between the MonHost and the Archive Server. This table consists of only the IP address and the name of the MonHost.

monhosts	
PK	<u>id</u>
	monHostName MonHostIP

Figure 3: monhosts Table on Archive Server

3.2.10 Summary Tables

The summary tables consist of data that has been processed and is ready for user to access.

summarytable	
PK	<u>id</u>
	MetricName
	ToolName
	MonHostIP
	MonNodeIP
	value
	MetricUnit
	Intestamp

Figure 4: Summary Table Format

It defines a metric value against its most important attributes. Summary tables exist on Monitoring Hosts and Archive Servers. The idea is to let the summary tables handle the Select queries to fetch metrics while using the other tables for temporary storage and tests.

3.2.10.1 Summary tables on monitoring hosts

The Monitoring Hosts consist of two kinds of Summary Tables. These are:

3.2.10.1.1 Raw summary tables

The format for naming of these tables is "summarytable[month][year]". These tables contain raw metrics for one month time period. These are updated after every hour by the application named "LocalBackup". These tables provide the primary data storage for all metrics; since the data table gets flushed after every day.

3.2.10.1.2 Summary tables with hourly aggregated values

The format of naming these tables is "summarytablebyhour[year]". These tables are formed by aggregating metric values for the last hour; using the aggregation scheme provided by the user. This scheme is discussed in detail in the coming sections. This table is again update after every hour by the application named "AggregatedSummarizationControlCenter". This table will contain fewer records and hence is formed for a complete year. The number of entries in one such table will be

$$n \times 24 \times 365$$

where n is the set of metrics with distinct combination of Tool, Metric and Path.

3.2.10.2 Summary tables on archive servers

The Archive Server consists of three types of summary tables; that are:

3.2.10.2.1 Summary tables with raw data:

This table consists of Raw Metric Data collected from all the Monitoring Hosts falling under an Archive Servers domain.

The format for naming of this table is "summarytable[month][year]". A server client model is used to transfer raw test data from the MonHost to the Archive Server.

On Archive server a server is always listening on a port. The value of this port is taken from epm-arc.config on archive server and epm-mon.config on MonHost. This table consists of the maximum number of entries as it contains Raw Data and that too for multiple Monitoring Hosts.

3.2.10.2.2 Summary table with hourly aggregated data

The format of this table is again "summarytablebyhour[year]", it is updated after every hour and the application doing so is "AggregatedSummarizationControlCenter". This table consists of hourly aggregated values but for all the Monitoring Hosts falling under an Archive Server.

3.2.10.2.3 Summary table with daily aggregated data

The format of naming this table is "dailysummarytable[year]", it is also updated after every hour by the application "AggregatedSummarizationControlCenter". This table consists of metric values aggregated after a day interval. The values from this table are taken from Raw Summary Table on the Archive Server. Hence if lets say a test is conducted just three times in a day then its value may be missed in the hourly aggregation but its result will be reflected in the daily aggregation table.

3.3 SCHEDULING

The basic aim of scheduling is to run the tests on all MonHosts in such a way that the increase in traffic does not corrupt Test results.

Currently EPM schedules tests independently on every MonHost. The idea is to group similar tests in a group, and then to identify which groups are to be run alone. Within each group, tests run in a sequence. If a group is to be run alone, all other groups are suspended and only the current group is allowed to run, this ensures that the group that is expected to get affected by other tests gets minimum interference from other tests.

As an example consider four tests T1, T2, T3, T4 and T5. Let's say T1 and T2 form a group G1, while T3 and T4 form G2, T5 runs independently and is placed in a group G3. G1 and G2 can run simultaneously, while G3 must run alone to ensure minimum interference from other groups. Within each group each test is given a sequence id, which indicates the sequence of running tests within each group.

Each group is then assigned a separate thread. Within each thread tests are run in a sequence according to the sequence number allocated to them. If however a group is encountered that has to run alone then all other threads are suspended till the time this thread with the run alone group has finished all it's tests.

Considering the above example, G1 is assigned thread Th1, G2 is assigned a thread Th2, and G3 Th3. Right at the start of every test a variable is checked to see if there is a group thread that wants to run alone, if yes this thread would simply go to sleep.

So when Th3 starts it set a variable to indicate that it needs to run alone, when Th1 and Th2 before the start of their next test find this variable to be set they simply go to sleep. Th3 runs, and completes its execution; it then set the variable to false, now when Th1 and Th2 wake up and find the variable to be false they simply resume their execution.

This ensures that on a MonHost, no test interferes with each other. Although this is not the best approach to this problem, but it provides a basic mechanism that can be improved to

decentralize the process of scheduling and form a protocol that can help to schedule tests between MonI hosts to ensure best results.

IMPLEMENTATION

This chapter contains implementation details of EPM; it consists of description for some important applications, the web interface to access the data, and details on how to deploy nodes and servers.

4.1 TESTER

The tester application performs the basic operation of conducting a test, extracting information and storing results in the Data table.

Tester starts by collecting information from configuration file named `epm-mon.config`.

```
epm:/home/epm # cd /etc/epm/
epm:/etc/epm # ls
aggregator.config  epm-arc.config  epm-mon.config
epm:/etc/epm # cat epm-mon.config
Archive Server IP =202.125.157.206
Archive Server Port=2000
MonHost JDBC URL =jdbc:mysql://localhost/epmmonhost1
MonHost JDBC User Name =root
MonHost JDBC Password =Vision2020
epm:/etc/epm #
```

Figure 5: `epm-mon.config` file format

This file contains entries such as Archive Servers IP, Monitoring Hosts, JDBC URL, JDBC user name and password etc. Values after being extracted from this file are used to connect with the database on the monitoring host. A loop then starts which ensures the application keeps on running, forever. Time is now checked, if it between 50th and 59th min of any hour, the application thread goes to sleep, it rechecks the min value after every 10 secs. As soon as the condition falsifies, the application connects to the database and collects Data necessary for conducting the Tests. All groups are placed in an array of "groupTest" objects. "groupTest" class extends Thread class and implements its run method; it also contains an array

with Test data. The constructor forms connection to the database, and the thread is made a daemon thread. For the group variable "isalone" is checked if it is set to 0, this thread is started. if however it is 1, class variable "TestAlone" is set to true and then this thread is executed. Every thread, before running a test, checks this variable and sleeps if "TestAlone" is true and it was not the one setting it true. After the thread, wanting to execute alone, has finished it sets "TestAlone" to false and all other threads can resume their tests. Inside the Thread for every test the "clientip" string in the argument field is replaced by the monitored node's IP. The Tool is executed and the result of this test is stored in a string with all line breaks replaced by "::-:". A tool's output can be extracted in two ways, either the output on the console or a file created by tool. Field "readResultFromFile" determines which methodology to use, if it is set to 0, output on the console is used, however if it is set to 1 then the entry in field "filePath" is used. This path identifies the location of the file where test results are stored. In both cases the output is read as a string and all line breaks are replaced by "::-:". Now if the "isRegular" field is set, the "matches" method of string class is used and this matches the "RegularExpression" with the result string and returns the result of this matching. This result is then stored in the database against the "TestID" and current "timestamp". However if it is set to 0, the application checks the value of "InformationExtractorClass" from the database, and loads the class that implements the "AbstractDataExtractor" abstract class. Afterwards the "processData()" method is called which takes the result string as input and produces a string as output. This string represents the metric to be stored and is stored in the database against the "TestID" and "timestamp".

4.2 DATA ARCHIVAL

The Data Archival applications, share a MonHost's data with its Archive Server. This consists of two applications, a server side application and a client sided one. The server side application or listens on a port as directed in the configuration file "epm-arc.config".

```
epm@epm:/etc/epm> cat epm-arc.config
Archive Server IP =127.0.0.1
Archive Server Port =2000
Archive Server JDBC URL =jdbc:mysql://localhost/epmarchiveserver1
Archive Server JDBC User Name =root
Archive Server JDBC Password =Vision2020
```

Figure 6: epm-arc.config file format

The client collects data for last hour and then connects to the Archive Server on the port entry, indicated in the configuration file epm-mon.config. It also takes Archive Server's IP address from the same file. The Archive server on receiving a connection first checks the "monhosts" table to find the client pc. If the client pc is a MonHost allowed by the Archive Server only then the server accepts data from the client. The client sends data for the last hour in form of queries appended in a single string; the server then separates these queries and runs them to store data in the database. This data is stored in the raw summary tables.

4.3 THE WEB INTERFACE

The Data Stored on the archive server is available via web interface. The main page consists of two parts the first is a list of summary tables present on the Archive Server, while second is a list of MonHosts.

[Home](#)

Welcome To EPM (under development)

Summary Tables

[click a link for details](#)

[dailysummarytable2003](#)

[summarytablebyhour2003](#)

[summarytablepl2003](#)

[summarytablepm2003](#)

[summarytablemay2003](#)

Monitoring Hosts

[click the link to open the monitoring host's website](#)

id	Name	Show Data
1	epmmonhost1	click here
2	epmmonhost2	click here

Figure 7: Main web interface

These MonHosts are the ones that can interact with the archive server and who fall under its domain. By clicking on the "click here" cell the MonHosts, website is opened in the frame below the table.

Monitoring Hosts

[click the link to open the monitoring host's website](#)

id	Name	Show Data
1	epmmonhost1	click here
2	epmmonhost2	click here

EPM MonHost 1

[Test Table](#) | [Metric Table](#) | [Nodes Table](#) | [Path Table](#) | [Tool Table](#) | [TestGroup Table](#)

Metric Table

id	Metric Name	Min Value	Max Value	Metric Value Type	Metric Unit
1	rtt	0	500	string	sec
3	loss	0	5	int	%

Figure 8: web interface- MonHost selected

Clicking on any of the buttons, the entries from these tables on the MonHost are shown under these buttons.

Clicking on a summary table would open a new page, where results from these tables can be viewed.

The screenshot shows a web interface with the following elements:

- A link labeled "Home".
- A text input field labeled "Timestamp" with a placeholder "%". To its right is the text "please put a % or a * to search for all values".
- A dropdown menu labeled "Timestamp (Human readable)" with a placeholder "%". To its right is the text "please select a time and its value will be propagated into the Timestamp field".
- A dropdown menu labeled "Select a MonHost to refine search" with a placeholder "%".
- Two buttons: "format timestamp" and "submit".

Figure 9: web interface-summary table selected

The Timestamp field on top takes in a UNIX timestamp value, while the one below shows timestamp in human readable form.

The screenshot shows the same web interface as Figure 9, but with the "Timestamp (Human readable)" dropdown menu open. The menu contains the following options:

- 21/6/2008--2:0:0
- 21/6/2008--3:0:0
- 21/6/2008--4:0:0
- 21/6/2008--5:0:0
- 21/6/2008--6:0:0
- 21/6/2008--7:0:0
- 21/6/2008--8:0:0
- 21/6/2008--9:0:0
- 21/6/2008--10:0:0
- 21/6/2008--11:0:0
- 21/6/2008--12:0:0
- 21/6/2008--13:0:0
- 21/6/2008--14:0:0
- 21/6/2008--15:0:0
- 21/6/2008--16:0:0
- 21/6/2008--17:0:0
- 21/6/2008--18:0:0
- 21/6/2008--19:0:0
- 21/6/2008--20:0:0

Figure 10: web interface-timestamp in human readable form

Selecting a value from this drop down menu would fill the Timestamp textbox with a corresponding UNIX timestamp.

[Home](#)

Timestamp: 1214038800 please put a % or a * to search for all values

Timestamp (Human readable): 21/6/2008—5:0:0 please select a time and its value will be propagated in the Timestamp field

Select a MonHost to refine search: %

format timestamp

Figure 11: web interface-timestamp selected

The MonHost drop down menu consists of all the MonHosts that shared their data in the last selected time period. This time period is the interval of the table, like hourly, daily, or monthly basis. Selecting a MonHost from this drop down menu displays all the MonNodes, this MonHost interacted with.

[Home](#)

Timestamp: % please put a % or a * to search for all values

Timestamp (Human readable): % please select a time and its value will be propagated in the Timestamp field

Select a MonHost to refine search: 192.168.5.119

select a monitored node to refine search: %

format timestamp

%
192.168.5.51

Figure 12: web interface-MonHost selected

Similarly selecting MonNode will display the metric entries and selecting metric will display tool entry.

[Home](#)

Timestamp: % please put a % or a * to search for all values

Timestamp (Human readable): % please select a time and its value will be pre-populated in the Timestamp field

Select a MonHost to refine search: 192.168.5.119

select a monitored node to refine search: 192.168.5.51

Select a Metric To refine search: rtt

select a Tool to refine search: ping

format timestamp

Figure 13: web interface-search criteria complete

Pressing the submit button now or during any phase displays the result based on search criteria.

[Home](#)

Timestamp: % please put a % or a * to search for all values

Timestamp (Human readable): % please select a time and its value will be pre-populated in the Timestamp field

Select a MonHost to refine search: 192.168.5.119

select a monitored node to refine search: 192.168.5.51

Select a Metric To refine search: rtt

select a Tool to refine search: ping

format timestamp (Day/Month/Year---Hour:minute:second)

select * from dailysummarytable2008 where MonHostIP='192.168.5.119' and MonNodeIP='192.168.5.51' and MetricName='rtt' and ToolName='ping'

Metric Name	Tool Name	Monitoring Host IP	Monitored Node IP	value	Metric Unit	timestamp
rtt	ping	192.168.5.119	192.168.5.51	1216157160	sec	1216184400
rtt	ping	192.168.5.119	192.168.5.51	1216157160	sec	1216184400

Figure 14: web interface-normal results

Checking the checkbox "format time stamp" would format the timestamp before displaying it.

Home

Timestamp % please put a % or a * to search for all values

Timestamp (Human readable) % please select a time and its value will be propagated in the Timestamp field

Select a MonHost to refine search 192.168.5.119

select a monitored node to refine search 192.168.5.51

Select a Metric To refine search rtt

select a Tool to refine search ping

format timestamp(Day/Month/Year---Hour:minute:second)

select * from dailysummarytable2008 where MonHostIP='192.168.5.119' and MonNodeIP='192.168.5.51' and MetricName='rtt' and ToolName='ping'

Metric Name	Tool Name	Monitoring Host IP	Monitored Node IP	value	Metric Unit	timestamp
rtt	ping	192.168.5.119	192.168.5.51	1216157160	sec	16/7/2008---1:0:0
rtt	ping	192.168.5.119	192.168.5.51	1216157160	sec	16/7/2008---1:0:0

Figure 15: web interface-results with timestamp in formatted form

4.4 DEPLOYMENT

During the whole design and development, easy installation and configuration of the whole system was kept in mind.

Core part of EPM consists of applications, databases and some configuration files. These Applications are JAR files, which require JRE version 1.2 and greater. The system has been tested with JRE version 1.4, 1.5 and 1.6.

The databases are built using MYSQL. There are two databases, one on the MonIhost and one on the Archive Server. The database on MonIhost has additionally two parts, one that is used by the Applications and the other that contains the performance measurements.

A short description for deploying EPM follows:

4.4.1 MonIhost

The process to set up a MonIhost is:

- Create MonIhost Database. Monthly Summary Tables can be ignored.
- Create a folder named epm and extract the rar file.
- Copy folder "epm" from epmMonhost to /etc/
- Fill the two configuration files with valid values
- Create a folder named logs
- Create a crontab using command crontab -e and enter:

```
01 * * * * java -jar [fully qualified path] / Tester / Tester.jar >> [fully qualified path] / logs /  
Tester.Report
```

```
52 * * * * java -jar [fully qualified path] / LocalBackup / LocalBackup.jar > [fully qualified  
path] / logs / BackupReport
```

```
54 * * * * java -jar [fully qualified path] / AggSumControlCenter /  
AggregatedSummarizationControlCenter.jar > [fully qualified path] / logs / AggReport
```

```
56 * * * * java -jar |fully qualified path| /  
BackupToArchiveServer_Client/BackupToASClient.jar |fully qualified path| / logs /  
ASClientReport
```

4.4.2 Archive Server

The process to set up an Archive server is:

- Create Archive Server Database. Monthly Summary tables can be ignored.
- Create a folder named epmServer and extract the rar file.
- Copy folder "epm" from epmArchiveServer in /etc/
- Fill the two configuration files with valid values
- Create a folder named logs
- Create a crontab using command crontab -e and enter:

```
58 * * * * java -jar |fully qualified path| / AggSumControlCenter /  
AggregatedSummarizationControlCenter.jar > |fully qualified path| / logs /  
AggSum.Server.Report
```

```
50 * * * * java -jar|fully qualified path| / BackupToASServerDist / BackupToASServer.jar >  
|fully qualified path| / logs / ASServerReport
```

FUTURE WORK and CONCLUSION

5.1 CONCLUSION

The current implementation of EPM serves as a preliminary prototype. It manages to achieve the main objective of automated testing, user defined metric extraction, managing data over different levels (on MonHost and the Archive Server), aggregating data and sharing network metrics through a web interface. At the same time, EPM is easy to deploy, configure and extend.

5.2 FUTURE WORK

Some additions to EPM that can actually help it improve are discussed as under:

As a first step, improved scheduling algorithm would be very much beneficial. The current algorithm reduces the time spent by an explicit test on an end node, what this means essentially is that a particular test that requires dedicated resources gets them only on the monitoring host. However for utilizing the network effectively and making sure that our tests return us accurate results, a scheduling scheme is needed to which all monitoring hosts within a domain can agree to. This scheme will involve the archive server to undertake the task of arranging tests in a way that critical tests are not bothered by other tests. So the MonHosts will have some preliminary data about tests that a user provides, this data is sent over to the archive server, so that it can arrange tests from all the MonHosts under its domain. The archive server then sends back the schedule and the MonHost will simply need to follow it to ensure optimum resource utilization.

The current web interface, is there to get the job done, and in no way the final solution. Sequence of graphs, new ways to share the data and improved information sharing are some of the additions that can be made to the current web interface.

In addition to these new better ways of integrating new Tools, extracting new metrics and overall provision of interfaces for EPM's management can be another viable addition. An

installation utility preferably integrated with the management utility would also be beneficial. This would make EPM easier to install and manage.

Currently the summary data is available only through a web interface. Provision of data through other means, like csv files, xml files and objects etc would also be beneficial.

REFERENCES

1. The PingER Project, Stanford Linear Accelerator Center (SLAC)
<http://www-iepm.slac.stanford.edu/pinger/>
2. Internet End-to-end Performance Monitoring (IEPM), SLAC
<http://www-iepm.slac.stanford.edu/>
3. perfSONAR
<http://www.perfsonar.net/>
4. Internet Performance Measurement and Analysis (IPMA)
<http://www.merit.edu/networkresearch/projecthistory/ipma/index.php>
5. Active Measurement Project (AMP)
<http://amp.nlanr.net/>

