Network Monitoring Tool

B. B. Meshram
Mumbai University, Mumbai, India
bbmeshram@roltanet.com

Mittal S. Bhiogade
Patni Computer Systems Ltd., Mumbai, India
mittalb@hotmail.com

T.R.Sontakke
S.G.G.S.C.E & T, Nanded, India
trsontakke@yahoo.com

Abstract

A Network Monitoring Tool “NETMON” is presented in this paper We have developed the tool that will Monitor the Network of any company say “XYZ” by performing certain Tests. NETMON will monitor the network by performing certain tests such as ping test, memory test, disk test, uptime test and URL test that would help to analyze where exactly the problem lies in the Network. The Results of the Tests are displayed in the forms of graphs that can be viewed on the browser. These Tests will produce the Results as the ouput, which will be stored in the database. Test Graphs will be generated from the Test Results stored in the database. These Test Graphs will help us to analyze where and at what time in the Network the congestion had occurred.

Keywords: Monitor, Collector, PerformanceAgents (PASP), Tests, Test Graphs, Network, Congestion.

Introduction

The primary pattern of interaction among the cooperating applications is known as the client-server paradigm. Client-Server interaction forms the basis of most network communication, and is fundamental because it helps us to understand the foundation on which distributed algorithm are built. The client-server paradigm uses the direction of initiation to categorize whether a program is a client or server. The term server applies to any program that offers a service that can be reached over a network. A server accepts a request over the network, performs its service and returns the result to the requestor. An executing program becomes a client when it sends a request to a server and waits for a response. Each time a client application executes, it contacts a server, sends a request, and awaits a response. When the response arrives, the client continues processing.

In Client-Server model applications, there are circumstances in which the externally offered load is larger that can be handled. Then if no measures are taken to restrict the entrance of traffic into the network, queue sizes grow indefinitely, the buffer space may be exhausted. When this happens, some of the packets arriving will have to be discarded and later retransmitted, thereby wasting the communication resources. As a result of which the actual network throughput decreases. [1,2,3]

In Client-Server applications the server provides certain services such as processing the database queries or sending out the current stock prices. The Client uses the services provided by the server to either display the database query results to the user or for making stock purchase recommendations to an investor. But sometimes the communication between the Client and Server is very slow. The reason for this would be network traffic bottleneck better known as “congestion”. Thus there are various problems in the network and one among them is congestion, this paper discusses how the congestion in the network can be located.

Material published as part of these proceedings, either on-line or in print, is copyrighted by Informing Science. Permission to make digital or paper copy of part or all of these works for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage AND that copies 1) bear this notice in full and 2) give the full citation on the first page. It is permissible to abstract these works so long as credit is given. To copy in all other cases or to republish or to post on a server or to redistribute to lists requires specific permission from the publisher at Publisher@InformingScience.org

A Network Monitoring Tool “NETMON” is presented in this paper. The paper contains the following contents: the solution of the problem, discusses the various tests, NETMON architecture and system design to monitor the network, NETMON tool algorithms, the experiments and test graph results and lastly we summarize the conclusion and future research directions.

Solution

The solution, which we suggest to analyze the Network is a Tool, called “NETMON” (Network Monitoring). It mainly comprises of three Programs viz. Monitor and Collector, which is to be installed on the Server machine and Performance Agents, which is to be installed on the Client machines.

The basic operation of NETMON is as follows, the Monitor (program installed at the Server machine) assigns the Tests that are to be performed by the Performance Agents (program installed at the Client machine), the Tests carried out at the Performance Agents produces Test Results as the output, which are sent back to the Monitor, the Monitor gives these Test Results to the Collector (program installed at the Server machine), the Collector stores the Test Results into database. The Tests Results can be viewed in the form of Graphs in the web browser, These Test Graphs helps us to analyze where and at what time there was the problem in the Network.

Tests

The Tests that are to be performed at Performance Agents are as follows

Uptime Test

This Test is used to determine the Uptime of the server i.e. the time interval between the current time of the server and the time when the server was loaded. This is to keep track of duration for which the server has been up.

Memory Test

This Test is used to determine the Memory Details such as total memory, free memory, swap in use and swap free memory. This is to keep track of how the Memory is utilized.

Disk Test

This Test is used to determine the Disk Details such as total disk space, used space, and free space. This is to keep track of how the Disk Space is utilized.

URL Test

This Test is used to determine the response time of the web server for a particular service

Ping Test

This Test is used to determine whether a particular service is up or not.

The Performance Agents will be performing the above mentioned Tests at the regular time interval which is configurable at the Monitor.

Netmon Architecture

The NETMON has five TIER architecture, which is shown in Figure1. Each TIER is explained below.

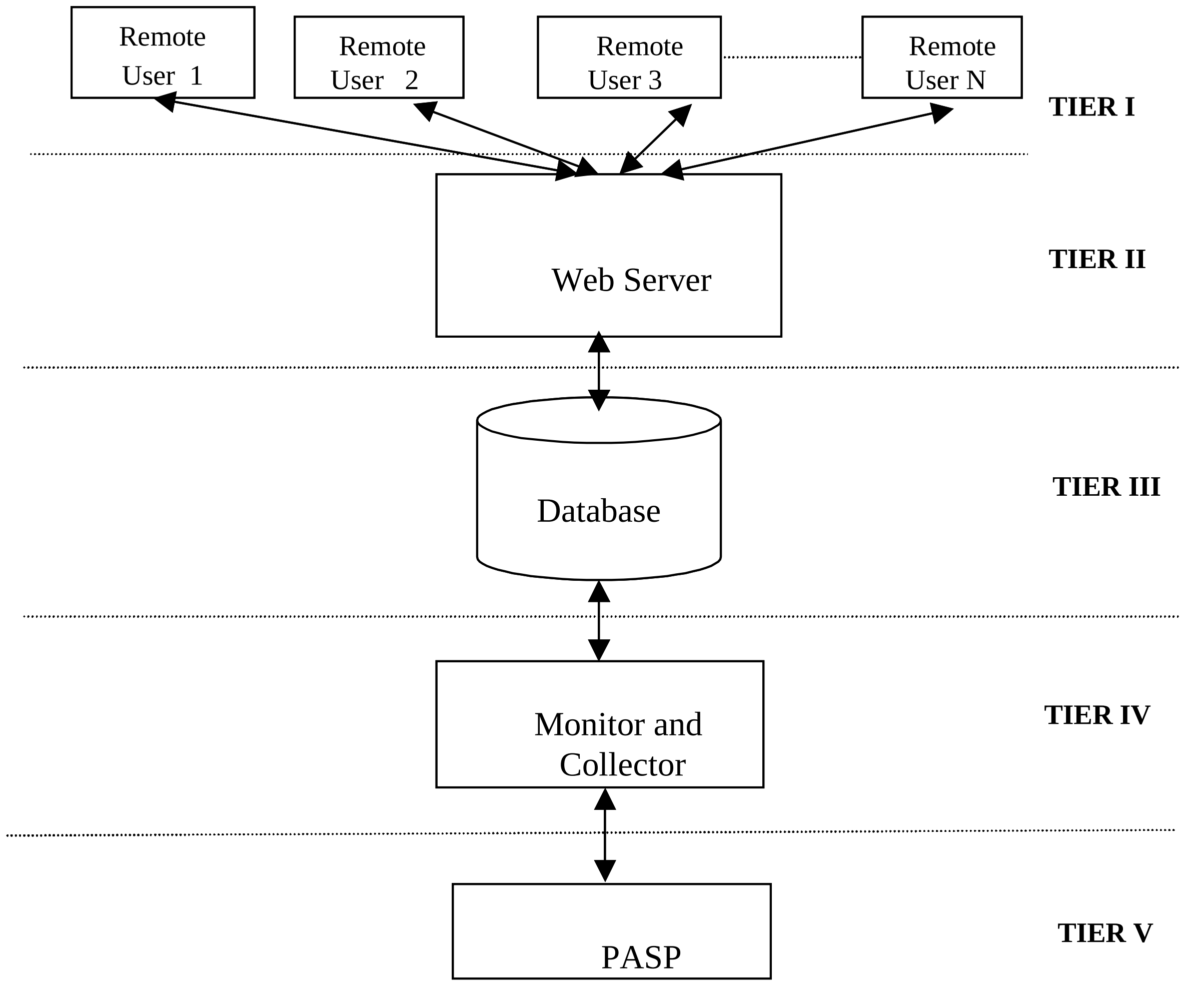

Figure 1: Five Tier Architecture of NETMON

Tier I

This is the Remote-User’s computer, from which the Remote User will view the TestGraphs in the Browser.

Tier II

This is the Application and Web Server Tier (Middle Tier) between the Remote User on Tier I and the Object Relational Database on Tier III.

Tier III

This is the database server, which stores the information in the tables. Data is accessed from the database in the form of queries.

Tier IV

This is the computer where the **“Monitor”** and “**Collector”** will be deployed. It is also referred as the Server machine.

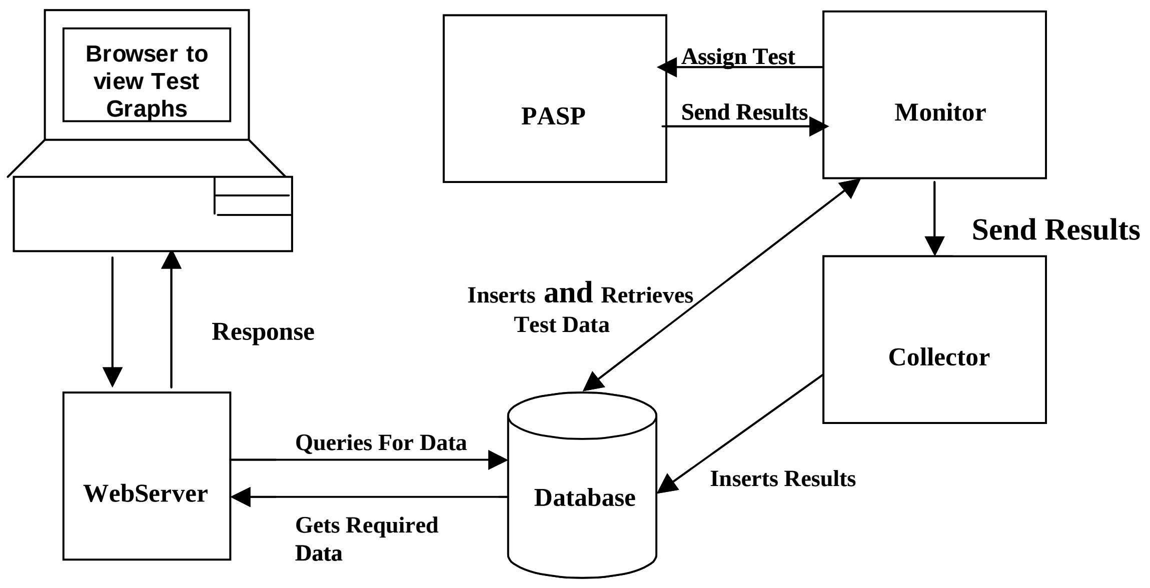

Figure 2: System Design of NetMon

Tier V

This is the computer where the Performance Agent **"PASP"** will be deployed. It is also referred as the Client machine.

System Design

System Design is the high level strategy for solving the problem and building a solution. System Design includes the decision about the organisation of the system into subsystems and major conceptual and policy decisions that form the framework for detailed design.

Monitor

Monitor is the program that has to be running on the server machine, its purpose is to assign the Tests to Performance Agents, which will perform these Tests and the Test Results that are generated by the Tests are sent back to the Monitor, and Monitor passes these Test Results to Collector.

Performance Agents

Performance Agent is the program that has to be running on the Client machine, its purpose is to perform the Tests that are assigned to it by the Monitor and to send the Test Results that are generated after performing the Tests to the Monitor.

Collector

Collector is the program that has to be running on the server machine, its purpose is to insert the Test Result data into database, which it receives from the Monitor.

Test Graphs

Test Graphs purpose is to give the Performance Analysis of the Test Result data collected and stored in the database in the form Graphs on the Web Interface. This is the Servlet based application that allows the

Servlet to query and retrieve the data from the database. It shows Test Graphs for all the Tests that are performed by Performance Agents.

System Operation

Step 1

Administrator starts the Monitor, Server socket is started which continuously listens at a particular port for Performance Agents Status Probe to get connected. At the same time Menu screen also appears which is running in a different thread. From the Menu screen, the Administrator can Add/Modify the details about the Performance Agents, Tests and Remote Users. During the allocation of the tests Performance Agents Status Probe may or may not be connected. If not connected then the tests are send to appropriate Performance Agents Status Probe as allocated by the administrator when Performance Agents Status Probe gets connected to Monitor.

Step 2

When the Performance Agents Status Probe is started, a socket is instantiated. The Server socket on the Monitor side accepts the connection. The Performance Agents Status Probe (PASP) enters login, password that is authenticated on the Monitor side. The Monitor checks if any Tests are allocated for that particular PASP. If yes then Monitor sends all the Test details to that PASP and PASP starts executing the Tests at the intervals specified to it in the Test details. Once the Test is executed the results are sent to the Monitor and gives the same to the Collector to insert it into Database.

Step 3

The Remote User will enter the Login and Password to view the Charts, after the successful validity of the Remote User, the graphs can be viewed by the Remote User.

Netmon Algorithms

NETMON has two main algorithms viz., for the Monitor and PASP. The algorithms are as follows:

Monitor Algorithm

This algorithm describes, how Monitor listens to the PASP which are trying to connect to it, the way the Monitor authenticates the PASP, sends the tests to PASP, and how Monitor receives the Results from the PASP, which are passed to Collector to be inserted into the database. [5,7,8,9]

Step1

Create a server socket, which listens continuously at some port say 8085 to determine whether any PASP is connected. MainMenu is displayed from which other screens can be invoked for the Monitor administration.

```
ServerSocket sockServer = new ServerSocket(8085);

GxsScrMainMenu objScr =new GxsScrMainMenu;
```

Step 2

If PASP is connected then create new socket for that PASP.

```
Socket sockClient = sockServer.accept()
```

Step 3

Store the reference of the socket connection and do the initialization.

```
Socket sockPASPClient = sockClient
initialise()
{
/* get the Data Input and Output Streams */
DataInputStream disInput = new DataInputStream(sockPASPClient.getInputStream());
PrintStream psOutput = new PrintStream(sockPASPClient.getOutputStream(), true);
/* get the Object Output and Input Streams  */
ObjectOutputStream oosOutput = new ObjectOutputStream(psOutput);
ObjectInputStream oisInput = new ObjectInputStream(disInput);
}
```

Obtain Login and Password details from PASP.

```
/* getting Login String */
     String strLoginInfo = disInput.readLine();
```

Step 4

Perform the check to determine whether Login and Password details provided by PASP are valid or not. This is achieved by the method authenticateLogin(), which return a boolean indicating whether the PASP is valid or invalid. If the Login and Password details of the PASP are invalid then the connection between the Monitor and PASP is closed.

```
        if(authenticateLogin(strLoginInfo))
        {
          psOutput.println("LOGIN_SUCCESS");
          blnValidated = true;
        }
        else
        {
          psOutput.println("LOGIN_FAILURE");
          blnValidated = false;
          sockPASPClient.close();
      System.out.println("Connection with client closed");
}
```

Step 5

Whereas if the Login and Password details are valid then, Monitor send the string “LOGIN_SUCCESS” as a token to the PASP, indicating that PASP connected is valid PASP. Tests that are to be performed by the connected PASP are sent to that PASP.

```
/* sending token to indicate that test are being sent from Monitor to PASP */
     psOutput.println("SENDING_TESTS");
sendTests(Integer intPASPId)
{
 /* fetch the Test Details related to the PASP from the database  */
 String strSQL = ”SQL statement that fetches the Test Details from the database”
/* selectQuery() method returns the vector of Test Data, by taking the SQL statement as the input
parameter */
Vector vecData = selectQuery(strSQL);
       if((vecData == null) || (vecData.size() == 0))
       {
         return;
  }
  else
 {
GregorianCalendar gcCal = new GregorianCalendar();
Vector vecResult = new Vector();
Iterator itData = vecData.iterator();
while(itData.hasNext())
{
  String strTestName = (String)itData.next();
  String  strJarName = (String)itData.next();
  String  strMainName = (String)itData.next();
  int intInterval = ((Integer)itData.next()).intValue();
  int intAppId = ((Integer)itData.next()).intValue();
  String strTestData = (String)itData.next();
  int intAllocationId = ((Integer)itData.next()).intValue();
  vecResult.add(new SendTestPASP(strTestName, strJarName, strMainName,
                 intInterval,intAppId, intPAId.intValue(), gcCal, strTestData,intAllocationId));
         } /* End-of-While */
psOutput.println("SENDING_TESTS");
/* sending the test data stored in the vector vecResult */
          oosOutput.writeObject(vecResult);
 }/* End-of-else*/
}
```

Step 6

The Monitor receives the results of the test performed at the PASP, these results are given to Collector, which inserts these results into database.

```
          processResults()
          {
        /* got the Result object  from PASP  */
            TestResults objTestResults = (TestResults)oisInput.readObject();
            /* sending the token to PASP indicating Monitor received the Result from the PASP */
                psOutput.println("TEST_RESULTS_RECEIVED");
               if(objTestResults!=null)
               {
         /* giving the objTestResult object to the Collector , to be inserted into the database */
                    Collector.processResults(objTestResults)
               }
```

Pasp Algorithm

This algorithm describes how the PASP connect to the Monitor, how does the PASP perform the Login to the Monitor, in case of incorrect Login the connection between the Monitor and PASP is closed, whereas after successful Login authentication it receives the tests from the Monitor and starts performing the tests, and how the Results of the tests are sent back to Monitor. [5,7,8,9]

Step 1

Creates the socket connection to the server

```
      serverConnection()
      {
Socket objSocket = new Socket(InetAddress,8085);   // InetAddress is the ipaddress of the Monitor
      DataInputStream input = new DataInputStream(objSocket.getInputStream());
      PrintStream output = new PrintStream(objSocket.getOutputStream(), true);
      ObjectInputStream objInput = new ObjectInputStream(input);
      ObjectOutputStream objOutput = new ObjectOutputStream(output);
      }
```

Step 2

Performs the Login to the Monitor, by entering the Login and Password details from the Login screen. These Login details are sent to the Monitor. If the Login details are incorrect, the message “Invalid Login” is displayed at the PASP. Whereas, if the Login details are correct PASP receives the Tests, it has to perform.

```
sendLogin()
{
 /* gets the login */
String strLogin=objLogin.getLoginPassword();
            if(strLogin != null)
            {
               /* Login details sent to Monitor */
               output.println(strLogin);
              /* login sucess or failure */
                strResult = input.readLine();
          / * if valid login then proceed   */
                 if(strResult.equals("LOGIN_SUCCESS"))
                 {
                    String strFromServer = input.readLine();
                    if(strFromServer.equals("SENDING_TESTS"))
                    {
                      vecTestName=(Vector)(objInput.readObject());
               } /* End-of-SendingTest */
            } /* End-of-LoginSuccess */
}  /* End-of-Login */
       else
       {
           /* Message displaying Invalid UserName/Password */
           objPopUp.showMsg("Invalid Username/Password");
}
```

Step 3

After the Tests are received successfully by the PASP, PASP will extract all the information regarding the Tests this is done by the method parseObject(), PASP sends the token to the Monitor indicating that the Tests are received successfully.

```
       parseObject(Vector vecTestName)
      {
        Vector v = new Vector();
        for(int i=0;i< vecTestName.size();i++)
        {
            v.addElement(((SendTestPASP) vecTestName.elementAt(i)).getTestName());
            v.addElement(((SendTestPASP) vecTestName.elementAt(i)).getJarFileName());
            v.addElement(((SendTestPASP) vecTestName.elementAt(i)).getMainFileName());
            v.addElement(((SendTestPASP) vecTestName.elementAt(i)).getTestInterval());
            v.addElement(((SendTestPASP) vecTestName.elementAt(i)).getTestData());
            v.addElement(((SendTestPASP) vecTestName.elementAt(i)).getAllocationID());}
       }
         /* token send from PASP to Monitor */
         output.println("TESTS_RECEIVED_SUCCESS")
      }
```

Step 4

PASP starts performing the tests as specified by the Monitor. The results of the tests performed at the PASP are sent to Monitor.

```
storeResults()
{
            /* This class would create the appropriate result object */
     SetResult objSetResult = new SetResult();
     /* returns the appropriate result object */
     Test objTestResult=objSetResult.getTestObject();
             /* token send from the PASP to Monitor , to indicate that test results are being sent */
             output.println("SENDING_TEST_RESULTS");
             /*sending result object to monitor  */
             objOutput.writeObject(objTestResult)
}
```

Experiments And Test Graph Results

We have used NETMON tool to monitor the network of the company “XYZ”, the results of this experiment is shown in the form of Test Graphs. The Test Graphs are explained below,

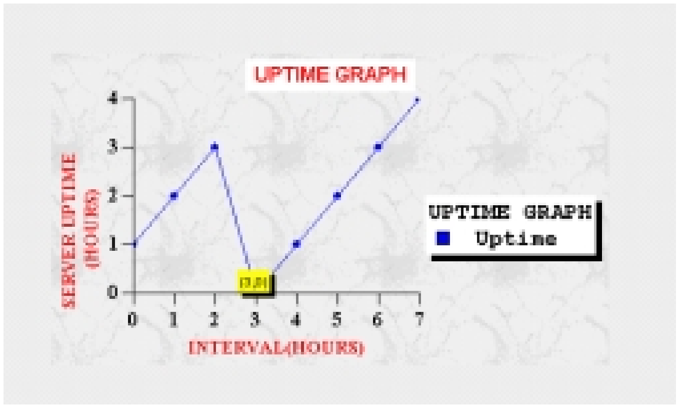

Figure 3: Uptime Test Graph

Uptime Test Graph

From Figure 3, it can be seen that on X-Axis there is Interval in “Hours” and on Y-Axis there Server Uptime in “Hours”. From the above graph it can be seen that at “0th ” Hour the uptime of the server is “1” Hours and at “1st ” Hour the uptime of the server is “2” Hours and at “3rd ” Hour the uptime of the server is “0” Hours, in the Figure 3 it can be seen that, the “yellow label (3,0)” indicates that on “3rd ” Hour the server was down for some reasons. Thus from Figure 3 it was possible to determine when the server was down.

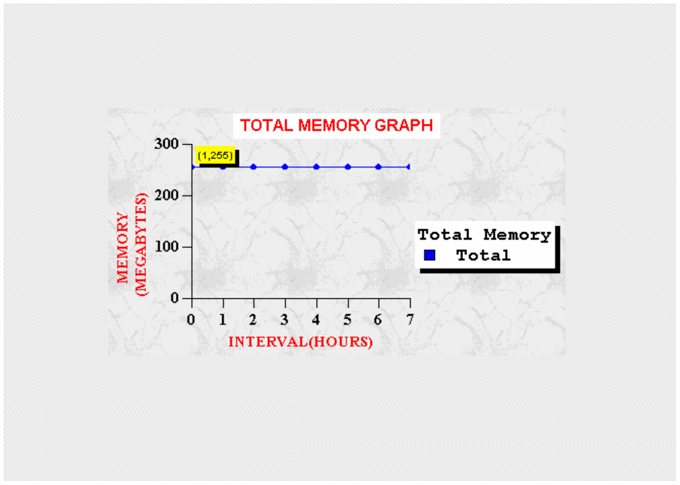

Figure 4: Total Memory Test Graph

Memory Test Graphs

Total Memory Graph

From Figure 4, it can be seen that on X-Axis there is Interval in “Hours” and on Y-Axis there is Memory in “MEGABYTES”. In Figure 4 it can be seen in the “yellow label (1,255)” indicating that at “1st ” Hour the Total Memory was “255MEGABYTES”. Thus from Figure 4 it is possible to determine how the Total Memory is utilized.

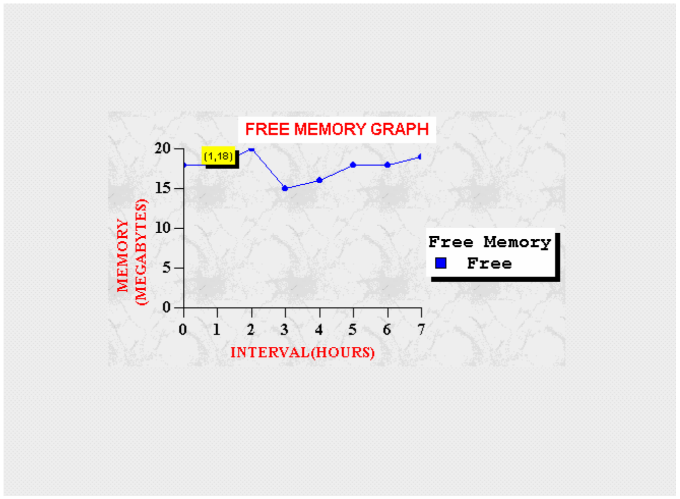

Figure 5: Free Memory Test Graph

Free Memory Graph

From Figure 5, it can be seen that on X-Axis there is Interval in “Hours” and on Y-Axis there is Memory in “MEGABYTES”. In Figure 5 it can be seen in the “yellow label (1,18)” indicating that at “1st ” Hour the Free Memory was “18MEGABYTES”. Thus from Figure 5 it is possible to determine how the Free Memory is utilized.

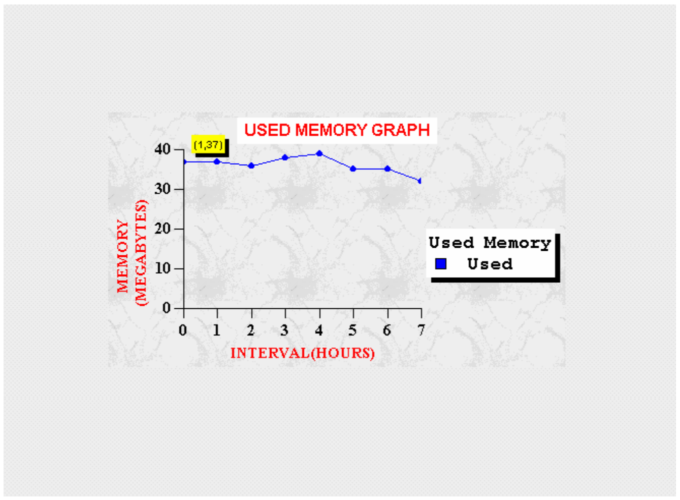

Figure 6: Used Memory Test Graph

Used Memory Graph

From Figure 6, it can be seen that on X-Axis there is Interval in “Hours” and on Y-Axis there is Memory in “MEGABYTES”. In Figure 6 it can be seen in the “yellow label (1,37)” indicating that at “1st ” Hour the Used Memory was “37MEGABYTES”. Thus from Figure 6 it is possible to determine how the Used Memory is utilized.

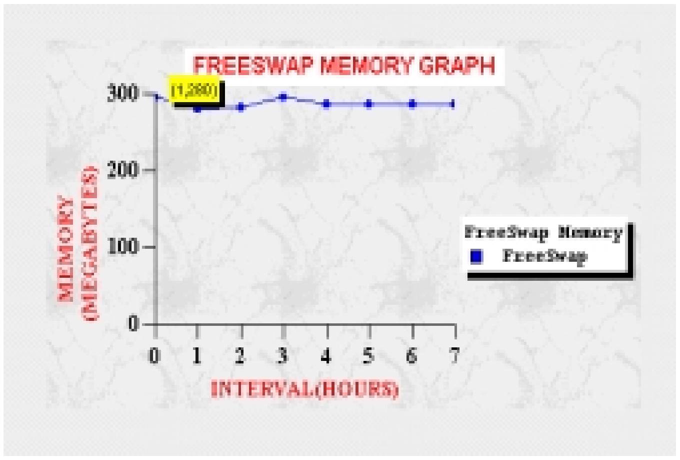

Figure 7: FreeSwap Memory Test Graph

FreeSwap Memory Graph

From Figure 7, it can be seen that on X-Axis there is Interval in “Hours” and on Y-Axis there is Memory in “MEGABYTES”. In Figure 7 it can be seen in the “yellow label (1,280)” indicating that at “1st ” Hour the FreeSwap Memory was “280MEGABYTES”. Thus from Figure 7 it is possible to determine how the FreeSwap Memory is utilized.

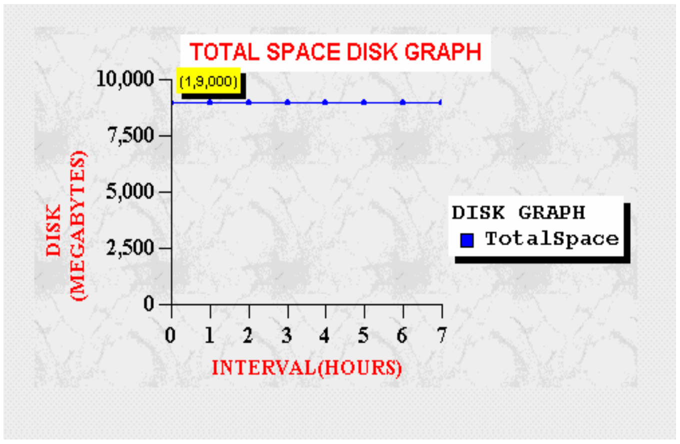

Figure 8: Total Space Disk Test Graph

Disk Test Graphs

TotalSpace Disk Graph

From the Figure 8,it can been seen that on X-Axis there is Interval in “Hours” and on Y-Axis there is Disk in “MEGABYTES”. In Figure 8 it can be seen in the “yellow label (1,9000)” indicating that at “1st ” Hour the Total Disk Space was “9000MEGABYTES”. Thus from Figure 8 it is possible to determine how the Total Disk Space is utilized.

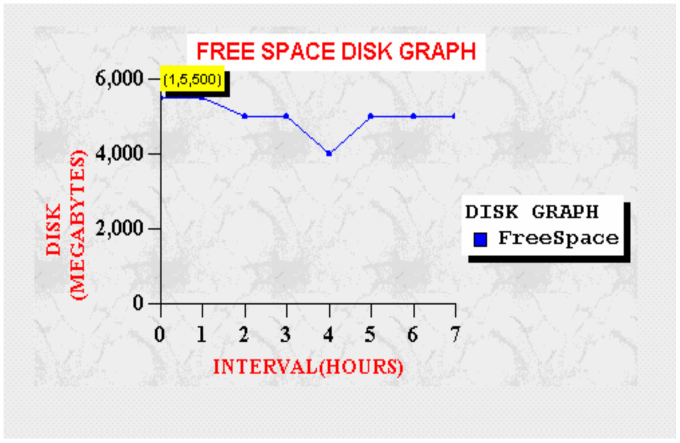

Figure 9: Free Space Disk Test Graph

FreeSpace Disk Graph

From Figure 9, it can be seen that on X-Axis there is Interval in "Hours" and on Y-Axis there is Disk in "MEGABYTES". In Figure 9 it can be seen in the "yellow label (1,5500)" indicating that at "1st " Hour the Free Disk Space was "5500MEGABYTES". Thus from Figure 9 it possible to determine how the Free Disk Space is utilized.

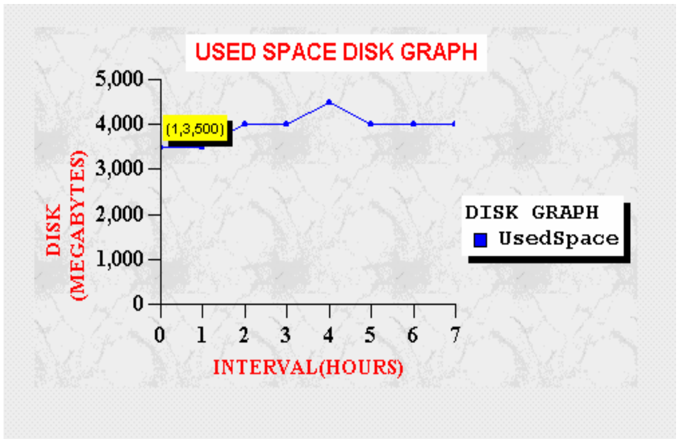

Figure 10: Used Space Disk Test Graph

UsedSpace Disk Graph

From Figure 10, it can be seen that on X-Axis there is Interval in "Hours" and on Y-Axis there is Disk in "MEGABYTES". In Figure 10 it can be seen in the "yellow label (1,3500)" indicating that at "1st " Hour the Used Disk Space was "3500MEGABYTES". Thus from Figure 10 it possible to determine how the Used Disk Space is utilized.

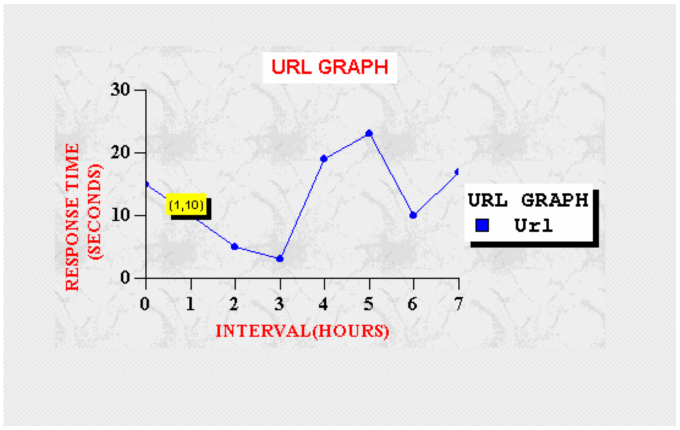

Figure 11: URL Test Graph

URL Test Graph

From Figure 11, it can be seen that on X-Axis there is Interval in "Hours" and on Y-Axis there is Response Time in "SECONDS". In Figure 11 it can be seen in the "yellow label (1,10)" indicating that at "1st " Hour the Response Time is "10 seconds". Similarly it can be seen that at "5th " Hour the Response Time is "23 seconds". Thus from the Figure 11 we can determine that at "5th " Hour response time taken is more and thereby from Figure 11 it possible to analyze the Response Time of the Web Server.

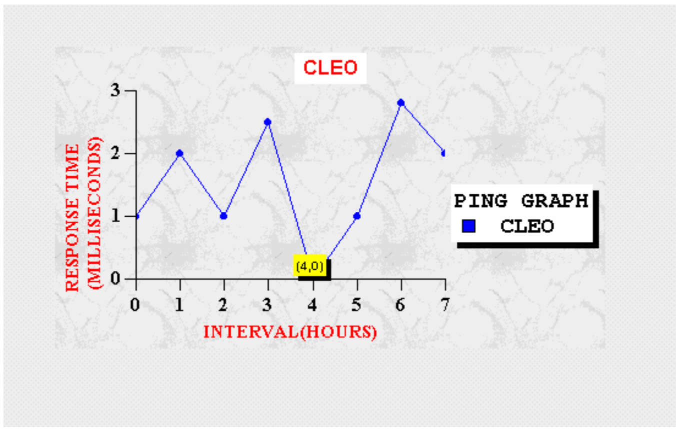

Figure 12: Ping Test Graph

Ping Test Graph

From Figure 12, it can be seen that on X-Axis there is Interval in “Hours” and on Y-Axis there is Response Time in “MILLISECONDS”. In Figure 12 it can be seen in the “yellow label (4,0)” indicating that at “4th ” Hour the Response Time is “0 seconds”. Indicating that particular service which is pinged is not available. Thus from the Figure 12 we can determine whether a particular service is available or not.

Thus with the help of Test Graphs we can locate the problems of the network and we can solve these problems manually.

Conclusion And Future Research Directions

Network Monitoring is very widely studied topic. In this paper, we have proposed a tool, NetMon to analyze the network. It performs Network Monitoring by running the different Tests to determine uptime of the server, memory space of the server, disk space of the server, to determine whether a particular service is available or not and to determine response time of the web server. Moreover this tool is developed using Java Technology at front-end and Object Relational Database Technology at back-end, as Java Technology is used it can be setup and run on any platform without making any change being made to its source code.

As the NETMON tool is used to monitor the network, in future this tool can be enhanced to monitor the web services. Today lack of performance guarantees in web services is often circumscribed by the fact that every thing is unpredictable and nothing can be guaranteed. A number of components may cause delays: the clients connection to its Internet provider, the provider’s proxy server, the web server application server at the target site, or content provider’s backend data server. Usually the network bandwidth is not the limiting bottleneck. Rather, the major reason for poor performance during peak hours is that requests suffer queuing delays at congested data server or gateways we have located this problem with the help of URL test. From a customer viewpoint, there should be money back guarantees: unacceptable performance results in no payment of compensation by the service provider. In 2005 all services that do not provide such guarantees should be out of business within their first month of operation. As a solution to this problem sometimes the best remedy is to make the analysis tractable. This would usually result in some performance losses, but the gain lies in predictability. With simpler building blocks absolute performance can be boosted by simply scaling up hardware resources. (i.e. memory disk . etc), while still being able to give strong performance guarantees.

The availability of a server is the probability that a client request at any arbitrary timeouts will find the server listening and ailing to process the request. The reason for temporarily being unavailable is transient failures and resulting down times. After a failure a server undergoes some recovery procedure, a “repair” and then resumes normal operation until the next failure occurs. In the long term, the availability of the server is given by the ratio

MTTF/(MTTF+MTTR) with MTTF denoting the mean time to failure and MTTR the mean time to repair. Note that periodic rebooting leads to occasional downtime and this adversely affects availability. Given that downtime, at the user-perceivable level is so expensive bold step towards continuous availability should have very high priority on our research agenda. The grand challenge for 2005 is to achieve less than one minute expected downtime per year which is equivalent to 99.9999 percent availabity, a Two orders of magnitude improvement of unavailability and a good approximation of truly non-stop services but the overriding goal of reaching 99.9999 percent availability requires significant progress on the engineering side. Furthermore software maintenance must be possible without interrupting system operation. For example it should be possible to upgrade to a new version of the operating system without having to bring down the database system on the same computer. Finally the key to satisfying performance goals even when some replicated components are down and have failed over to configure the overall system appropriately. Most importantly, the degree of replication for the data and process determine not only the absolute availability but also the effective performance, note that to overcome the problem of server availability, we have implemented the Ping test that checks the availability of the particular server continuously, this can be extended to achieve the same for Web Services.

Acknowledgements

The authors thank Dr. S.G.Bhirud, Prof. T.M. Bansod, Mr. Sachin Rao, Mrs. Sonal Desai, Mrs. Sona Meshram and Mrs. Vatsala Madke for their participation in several aspects of this work.

References

[1] D.E. Comer (1999), *TCP-IP Vol. I*, Third Edition. PHI.

[2] D.E. Comer & D.L.Stevens (1998), *TCP-IP Vol. II*, Second Edition, PHI.

[3] D.E. Comer & D.L.Stevens (1997), *TCP-IP Vol. III*, PHI.

[4] Patrick Naughton (1991), *Java 2 Complete Reference*, Addison Wesley.

[5] Sun Microsystems Java Documentation., "http://www.sun.java.com"

[6] Dimitri Bertsekas & Robert Gallager (1997), *Data Networks*, Second Edition, PHI.

[7] Matthew Siple (2000), *The Complete Guide to Java Database Programming*, Tata McGraw-Hill.

[8] C.S. Horstmann & G. Cornell (2000), *Core Java Fundamentals Vol I*, Sun Microsystems Press.

[9] C.S. Horstmann & G. Cornell (2000), *Core Java Fundamentals Vol II*, Sun Microsystems Press.

Biography

B. B. Meshram is currently an Assistant Professor of Dept. of Computer Technology in V.J.T.I, with the university of Mumbai.He has received B.E. (Computer Engineer) in 1991 from Marathawada university Aurangabad, M.E. (Electronics) in 1995 from Dr.BAMU Aurangabad He has taught various subjects like System Analysis & Design, Compiler Construction, Theoretical Computer, Science, Advanced Databases, Parallel computer Architecture, Object Oriented Analysis & Design, Computer Networks, Computer Graphics at graduate & Post Graduate level. He was the Chairman of National Level Computer Science Symposium (Interface 95) at V.J.T.I.Mumbai-19.He is the Life Member of CSI. His research areas are Networking, Object Oriented Programming & Object Oriented Databases.

Mittal. S.Bhiogade is currently working as Software Engineer with Patni Computer Systems Ltd., he has received B.E. (Computer Engineer.) in 1999 from Sardar Patel College of Engineering. (S.P.C.E), Mumbai University Maharashtra, his research areas are Networking, Socket Programming & Object Oriented Programming.

Dr. T.R.Sontakke received the Doctoral degree from I.I.T. Bombay in 1980 and was a Professor & Head of dept. of computer & electronics. in S.G.G,S.C.E.& T. Nanded for 15 years . He has guided many hardware and software projects at graduate & postgraduate levels. He has published many papers in various Nationals and international journals. Presently he is the principle and Secretary of the said Engineering college. His research areas are Networking, object-oriented technology, Databases & System software.