ConTest: A GUI-Based Tool to Manage Internet-Scale Experiments over PlanetLab

Basheer Al-Duwairi
Jordan University of Science and Technology/Department of Network Engineering and Security, Irbid, Jordan
Email: basheer@just.edu.jo

Mohammad Marei, Malek Ireksossi, and Belal Rehani
Jordan University of Science and Technology/Department of Computer Engineering, Irbid, Jordan
Email: {mymarei, malek.erq, belalrehani}@hotmail.com

***Abstract*— PlanetLab is being used extensively to conduct experiments, implement and study large number of applications and protocols in Internet-like environment. With this increase in PlanetLab usage, there is a pressing need to have efficient way to setup and mange experiments. This paper proposes a graphical user interface-based tool, called ConTest, to setup and mange experiments over PlanetLab. ConTest enables PlanetLab users to setup experiment and collect results in a transparent and easy way. This tool also allows different measurements for different variables over the PlanetLab network. The paper discusses the design and implementation of ConTes and shows different scenarios.**

***Index Terms*—Internet measurements, PlanetLab. API, Networking, GUI.**

I. Introduction

PlanetLab is an open shared platform for developing, deploying, and accessing planetary scale applications [1]. Currently, it has more than 1100 nodes distributed all over the world. It allows its users to freely access the shared nodes, upload programs and execute them. It is used by many distributed applications developers all around the world for testing their applications (e.g., [2, 3, 4, 5]). It can also serve the purpose of having machines distributed all around the Internet to break the limit of locality and allow the users to perform their measurements on random nodes at random places to ensure generality and confidence in the obtained results.

PlanetLab is being used extensively to conduct experiments, implement and study large number of applications and protocols in Internet-like environment. These applications and protocols fall into different areas that include real-time measurements of delay, bandwidth, and security. With this increase in PlanetLab usage, there is a pressing need to have efficient way to setup and mange experiments. Having such tool would be a great assist not only for Planetlab community, but also to new researchers/users who find it difficult to explore and utilize this platform. In fact, several systems (as described in Section V) have been proposed to achieve that goal.

This paper builds on previous efforts in this direction and proposes a GUI-based tool, called ConTest, that allows users to visually manage experiments in Internet-like environment. In this regard, ConTest provides great deal of flexibility and simplicity to conduct experiments over PlanetLab. Using this tool, users have the ability to visually create the network topology they want simply by selecting PlanetLab nodes as they appear on the world map view. Also, it allows users to specify the role of each node (client, server, or proxy) and to generate different type of traffic (e.g., TCP, UDP, ICMP, etc.). In addition, ConTest has the capability to perform basic operations in sequence or in parallel. This include authentication, file upload, and command execution.

ConTest provides efficient and simple way to deploy and test applications over Planetlab. With this tool it becomes very easy to study the properties of Internet applications and protocols. It also becomes easy to measure their performance in terms of response time, delay jitter, packet loss, etc. With its ability in supporting application deployment and monitoring in Internet-Like environment such as PlanetLab, we believe that ConTest would promote research in this field and would significantly contribute to the increase of PlanetLab users because of its attractive features that hide many of the complications behind the scene.

The rest of this paper is organized as follows: Section II explains ConTest architecture. Section III discusses implementation details. Section VI discusses security issues. Section V discusses related work. Finally, conclusion and future work is presented in Section VI.

II. ConTest Architicure

The design and implementation of ConTest is based on modularity and usability concepts. In this regard, ConTest is composed of several modules with each module having specific functionality. In particular, ConTest has the following modules: Authentication Module, Command execution module, File management module, and Connection management module. What follows is a description of each module.

© 2012 ACADEMY PUBLISHER
doi:10.4304/jetwi.4.2.164-171

A. Authentication Module

Typically, users access PlanetLab nodes through *slices*. Basically, a slice is a multi-user account registered on the PlanetLab Central (PLC) to offer all the users of that account the same resources on the nodes which they add to their slice such that users of that account work on the same project and preserve node' resources of that slice. Accessing a PlanetLab node requires the user to have access to the slice to which the node belongs to. PlanetLab offers a web interface for the users to allow them to access their slices and see the nodes registered there. But it also offers an API that allows external programs to communicate with PLC databases through remote procedure calls (RPC) to retrieve users' account and node information. ConTest utilizes this API to perform authentication.

The authentication module is a program that communicates directly with the PlanetLab central API. It accepts the authentication information (email, PlanetLab password and slice name) from the main module. This information is then validated by the PLC. After that, it retrieves the slice info from the PLC and the node list registered on this slice, with all the necessary data about each node (ID, name, location, status, etc.) as illustrated in Fig. 1.

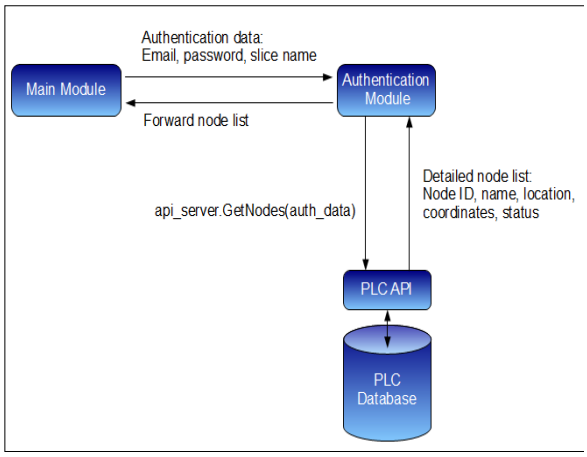

Figure 1: How the authentication module accesses the API to retrieve node list

B. Command Execution Module

The command execution module allows users to run their own custom commands on the nodes to which they have access (i.e., within their slices). Also, it is used in the connection management module to run the tests requested by the user. In order for the tool to communicate with the remote nodes and access their resources to perform the required test, and due to the security restrictions on the access to PlanetLab nodes, any connection to the nodes must be established through the secure shell (SSH) protocol. Therefore, command execution is done through SSH connections, where the command execution module connects to the desired nodes using the secure public key cryptography authorization, sends the command to the destination nodes and retrieves the output and error messages from each node.

C. File Management Module

Similar to the command execution module, file copying is done through SSH and using the secure copy function. After accepting the key pair identification, the files are uploaded to the destination nodes and their console dump is retrieved to ensure that the copying process was successful.

D. Connection Management Module

This module represents the operational core of the tool. In order for the user to perform their tests, they need to connect the nodes together using a testing program which they also need to upload. The connection management module does that for the user. Also, this module heavily utilizes the program capabilities. It uses a special class to setup the connections required by the user. The role of each node (e.g., client, server) is determined using the programmed connection algorithm. The connection management module communicates with the map interface on the user interface to help the user select their connection topology in a simple and efficient way. It also utilizes the file copying and command execution modules to upload the test files to the desired nodes, execute these files, retrieve the results from each node and output these results to the GUI.

The results associated with an experiment are stored to separate files for reference. Also some statistics (e.g., bandwidth, round trip time, etc.) are displayed to the users upon the execution of these experiments. Fig 2. shows the execution flow for the connection management module and its interaction with other modules in each step.

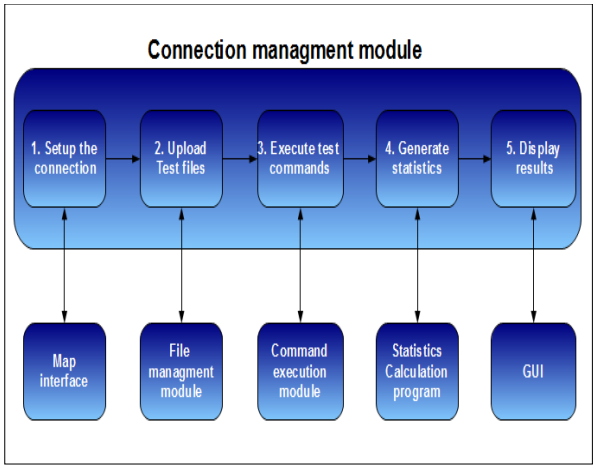

Figure 2: Execution flow in the connection management module

© 2012 ACADEMY PUBLISHER

The GUI represents the main program that links all other modules of the tool. The GUI module accepts user inputs and manages outputting results and error messages to the user in text file format and to the program log. It communicates with the authentication module to read node data and display them in table format, as well as identify and show their location on the world map using the retrieved coordinates from the authentication module. It also sends the user input to the command execution and file management modules and retrieves the console log from each module to display it in the program log or save it to separate files. Also, it integrates the connection management module with the command execution/file management modules to achieve the tests required by the user.

F. ConTest Classes

GUI class: The core of ConTest is the GUI module/class, which connects the other modules together and manages their interactions. This class is responsible for interacting with the user, I/O management between the sub-classes, and error handling. Most of the content of the main class concerns the graphical widgets displayed on the GUI, the initiating of the other modules and the management of their inputs and outputs. So we won't go deep into describing this class as it contains many auto generated elements (the graphical widgets). That leaves us with the sub-classes in the design, which are:

Copying thread class: The copying thread class is the class that represents the file management module. Instances of this class receive and store the essential data of the nodes to which the copying is going to be performed, as well as the data of the file to be copied to the nodes, this includes:

- Node name.
- Key file (for the ssh connection).
- Source file (the file to be copied).
- Destination directory on the remote node.

The instances run a *scp* (secure copy) command using the information passed by the main module, retrieves the console output from the remote nodes and relay this output to the GUI.

Execution thread class: Similar to the file copying class, instances of this class contain the data of the nodes which the command is going to be executed on:

- Node name.
- Key file.
- Command string.

Each instance of this class runs a *ssh* command using that information, which is passed to it by the main module, retrieves the console output from the remote nodes and relays this output back to the GUI.

Node view class: This class is mainly used to represent the nodes graphically, it also stores all the available data of the nodes, namely:

- Node ID.
- Node name.
- Location.
- Status.
- Coordinates (for display in the GUI's map view).

It also contains pointers to other nodes. These pointers are used in creating linked lists that help in constructing the required connections for the tests performed by the connection management module. Fig. 3 illustrates the relationship between the different classes; it also shows the relationship between the main module and the authentication module.

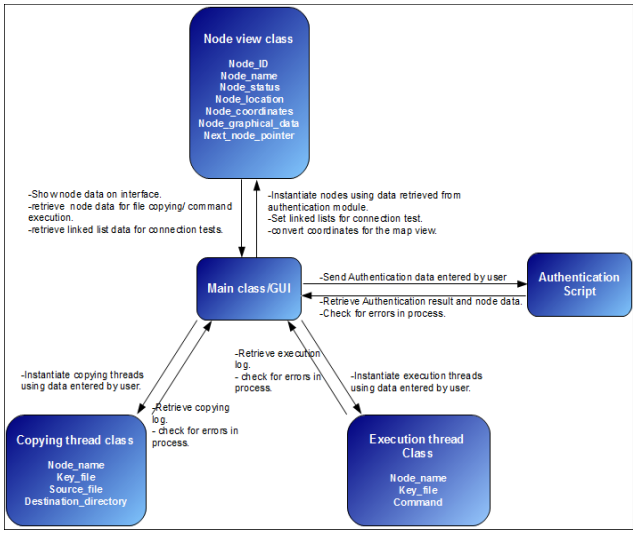

Figure 3: Relationship between different classes

III. IMPLEMENTATION DETAILS

A. Tools and Programs used to Implement ConTest

ConTest was implemented using Qt Qreator [6][7], a widely used IDE (integrated development environment) that helps programmers create powerful graphical tools, which provides an effective way to integrate an easy to use, simple graphical interface with the ability to perform a wide set of different operations (like accessing APIs, using the SSH commands, displaying map, etc.). Qt Qreator has a huge library of graphical and non graphical classes and functions that enable the programmer to do almost anything they want, helping them give their programs the maximum functionality. It uses C++ as the programming language, and its set of libraries makes the programming process fast and effective, and helps the programmer to focus on the core functionality of their program while it handles the graphical aspects.

In addition *Ping* [8, 9] and *Iperf* [10] have been integrated within ConTest to provide means to measure different aspects of network performance related metrics, such as round trip time and connection throughput. In ConTest, this can be done by starting two processes on the

© 2012 ACADEMY PUBLISHER

designated remote node, one measures the bandwidth using Iperf tool and the other measures the round trip time using the ping command. These processes then return their output through SSH to the program in order for it to record it locally and analyze it to get the required connectivity statistics about all the nodes of interest to the user.

B. Inter-process Communication

The main program, the GUI, constitutes the main process that initializes and connects other processes, thus the other sub-processes take their required running data from the GUI through an inter-process link, then they run their separate codes independently in the background and return the output to the main program through the same link.

When created, each sub-process gets initialized and sent its parameters, and then each process establishes a connection to the main process through two channels, the standard output channel (*stdout*), which communicates all outputs from this process to the main process, and the standard error channel (*stderr*) which transfers the error messages (if any) from the sub-process back to the calling process. Even when a sub-process needs to communicate with another sub-process, this communication needs to go through the main process as illustrated in Fig. 4.

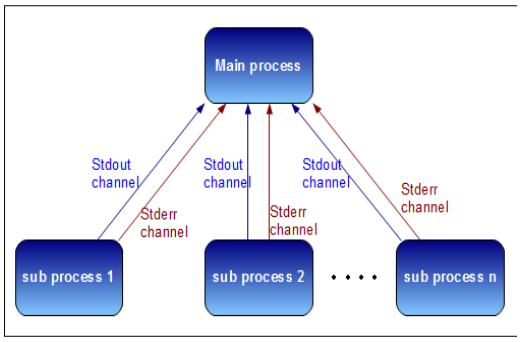

Figure 4: Process communication through channels

C. Multi-threading

Since the execution of commands and file copying through the SSH is a lengthy process, and since we need to do a lot of copying, command execution and file writing, along with the active interaction with the GUI, we had to program the tool in the form of threads. Threads are, essentially, child processes that execute concurrently. Each separate thread runs in parallel with the GUI, thus keeping the program responsive at all times and increasing the utilization of the machine resources.

Of course, these threads need to access the GUI to write their output through it to the user, however, multiple threads writing to the same object (i.e. the program log) is prohibited as it can cause access errors and even program crashes. So we had to use the concept of event/event handler or, as it is called in the Qt environment, signal/slot pair. When a thread has some output that it needs to write to the interface, it emits a signal to the GUI telling it that its output is ready. The GUI receives this signal through a specific slot and puts it in the queue of signals that wish to access the interface. This regulation prevents multiple access to one object at the same time and gives each thread its turn to write to the interface.

D. Authentication

As mentioned earlier, the PlanetLab Central has high security standards that restricts the access to their user and node databases, but fortunately, they provide the API to access their PLC database and retrieve the node data we require for our tool, thus we have programmed the separate authentication script. This script does the authentication through the API functions and using the user's data provided in the GUI (their slice name and PlanetLab registered email and password), and retrieves the detailed node list for that specific user. Also, the connection to the nodes themselves requires the use of the SSH protocol, and since SSH uses public key cryptography to authenticate to the remote computer, we had to ask the user for the key file which they have registered in their PlanetLab account.

Another issue with the SSH connections is that SSH protocol requires the key's password each time you connect to the remote host for confirmation. This is the default process and it is recommended as it prevents altering or theft the user's SSH key, but if the user needs to access multiple hosts at a time, it will make the connection process more redundant and time consuming. However, since this check can be disabled through the command options, we decided to give the user the liberty to choose whether to enable this confirmation or not, but disabling the check is not recommended as it may raise security warnings and risk the user's key security.

VI. ConTest Functionality and Usage

ConTest provides an easy and flexible way to setup and perform experiments over PlanetLab. This is done mainly through the *experiment manager interface*, where the user specifies experiment parameters and through the *connection tester interface* which establishes different connection schemes between nodes and measures some aspects of these connections. Before going into the details of these two interfaces we describe the way authentication is performed.

A. Authentication

The authentication dialog pops up at the start-up of the program, it requires the following data to enable you to use the device:

- Email: the e-mail which you have registered in your PlanetLab central account.

© 2012 ACADEMY PUBLISHER

- Password: the password of your account.
- Slice name: the name of a slice reserved for you on PlanetLab.
- Key file: the directory of your private key that is used to access the nodes through the secure shell (*ssh*).

Upon a successful authentication, user information will be displayed in the upper right corner; the number of nodes in the provided slice will be displayed in the program log. Also the details of the node list registered on the slice are displayed in the table view and the nodes themselves are represented on the map as shown in Fig.5.

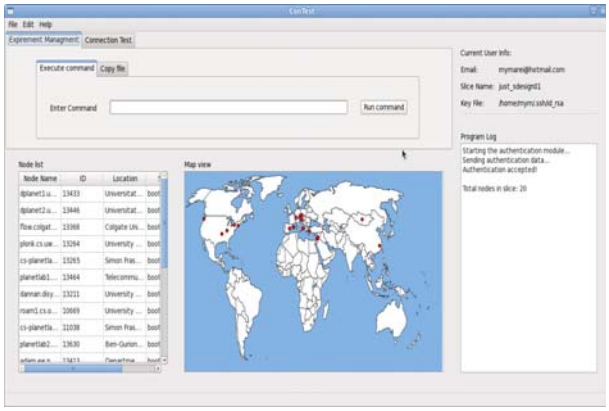

Figure 5: After successful authentication

B. Experiment Management

In this tab, user can execute commands on remote nodes or upload files to them. The basic functions that can be performed within this tab are:

- *Node selection:* Nodes can be selected by clicking on them in the node list table or by clicking on the nodes' icons on the map. Information about a node can be obtained by hovering the mouse pointer on it, this will display a tool tip showing the node details.
- *Command execution:* To execute a command, you simply select the nodes on which you want to run your command, insert the command text with all variables in the command text box, and then click the "run command" button. The output from each node will be displayed on the program log.
- Uploading Files: To upload files to nodes, go to the "copy file" tab, from there you can select the file which you want to upload using the "browse" button, you can also specify the destination directory on the remote nodes which you want to store your file in, after selecting the nodes you click "upload file" to start uploading your file. When uploading is done, the log will display the output from the remote consoles to verify that the process is successful, if there are no error messages then the file is uploaded.

C. Connection Testing

Connection testing tab provides an interface for users that allows them to test the connection between any nodes from the list of nodes within a slice. It also allows selecting the protocol of preference. The test results are displayed on the program log and stored to external files for documentation. The basic functions that can be performed within this tab are:

- *Test Protocol:* from here you should be able to select your has its own client, server and forwarding program. You can select the protocol from the "test protocol" drop down box. And you can also browse for your own programs if you have customized a code to be used with ConTest. To restore the defaults test programs, just press the "restore defaults" button.
- *The Map Interface:* In the connection tester, you can add nodes to test only through the map interface. This is done by clicking on a node using the left mouse button. As long as you keep selecting nodes using the left button you will keep the path open. Until you close it by clicking a node using the right mouse button, thus setting it as a server. Currently, you can't give a single node two roles (e.g. a client and a server at the same time).

After setting up your connection, you click the "run test" button to start your test, when the test is done, its results are displayed on the program log as shown in Fig.6.

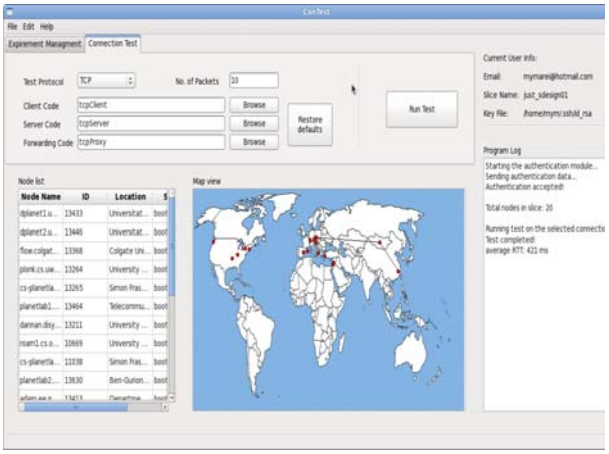

Figure 6: Adding nodes to a path, : The test results showing on the program log

D. Illustrative Examples

We consider two scenarios to show the functionality and flexibility that ConTest offers in setting up and executing experiments over PlanetLab. In the first scenario, we show how to use the tool to obtain information about the disk usage percentage on each node within a slice. In the second scenario, we show how to perform round trip time measurements between two nodes and how to assign roles for each node.

© 2012 ACADEMY PUBLISHER

Example 1-Retrieving Disk Usage Percentage: In this example, we show how to access multiple PlanetLab nodes simultaneously and perform a simple test to obtain the percentage of the disk space used on each node within a slice. In order to perform this test, a shell script responsible for obtaining the disk space usage has been uploaded on each node within our slice. This was done by selecting all nodes from the list in the node list view and uploading the shell script "*diskUsuage.sh*". To find the disk usage percentage on a node, the shell script was executed via the execute command tab where the path of the script "*testScript/deskUsage.sh*" is specified. Disk usage percentage is displayed in the program log area as shown in Fig. 7.

Example 2: Round Trip Time (RTT) Measurements

In ConTest, the connection test interface allows users to specify the role of each node in an experiment. In this regard, a node can be specified to be a client, a server, or a forwarding node (i.e., a proxy). Also, it allows them to upload the code that is suitable for the role of each node. Measuring the round trip time between two nodes is a simple example that we describe here to illustrate this functionality.

Fig. 8 shows the results of measuring RTT between a client (the green node) and a server (the pink node). It also shows the results of measuring RTT between the same nodes assuming that the traffic is forced to pass through another node (the proxy) represented by orange in this example. Such setup is commonly used for indirect communication and it can serve for testing triangular

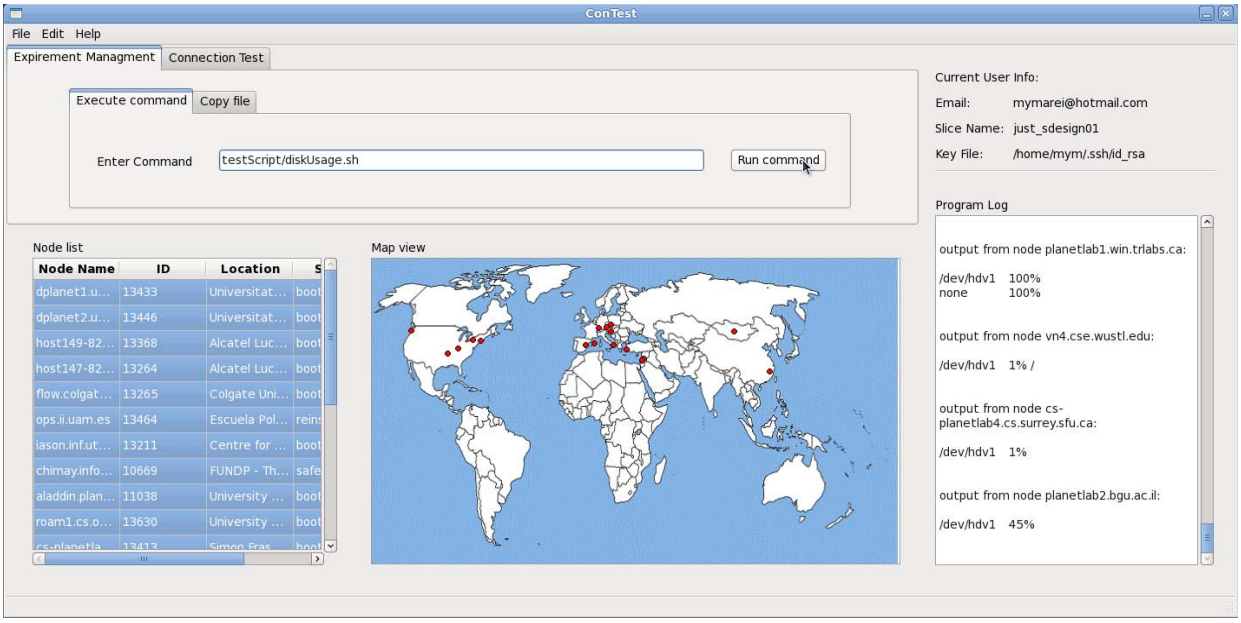

Figure 7: Using ConTest to obtain disk usage percentage of each node within a slice (Example 1)

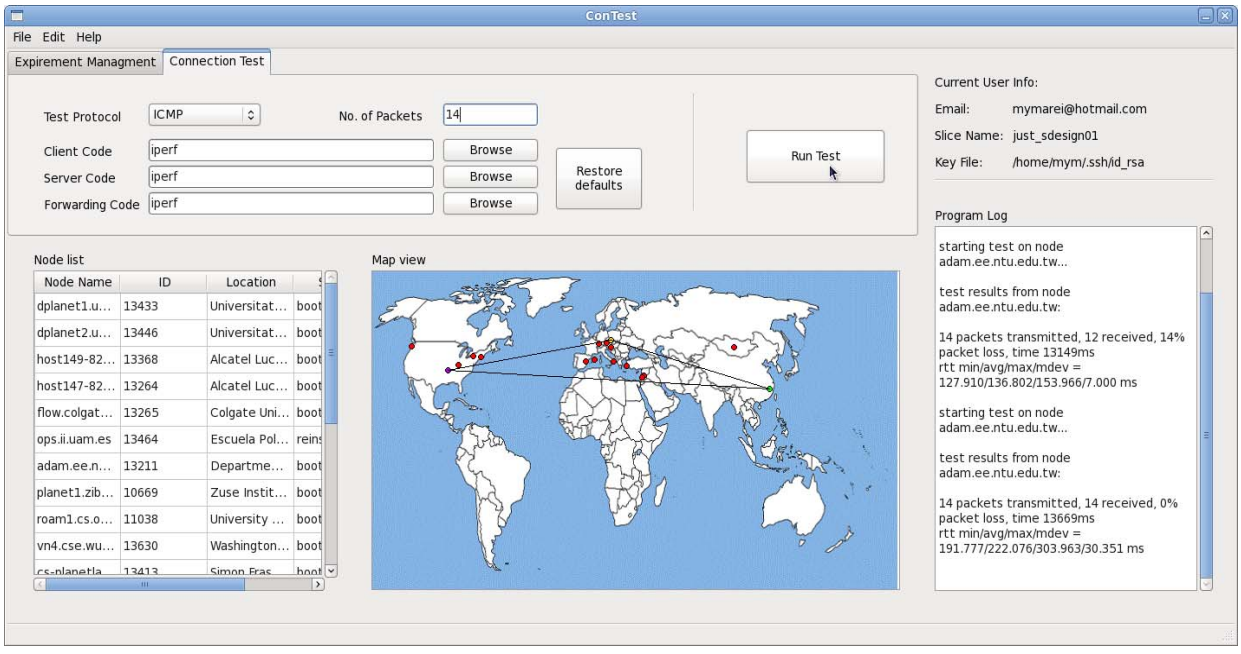

Figure 8. The results of measuring round trip time between two nodes (Example 2)

© 2012 ACADEMY PUBLISHER

routing or even studying some characteristics of fast flux networks which is becoming a very important problem [11][12].

It is important to emphasize here that using ConTest to perform such experiments over PlanetLab provides great deal of flexibility and saves a lot of time to setup and run the experiment and collect the related results. So instead of using the CLI to access each node individually to upload the files, execute commands or setting up tests, ConTest automates this process and saves the user's time and effort.

V. Related Work

Since the deployment of PlanetLab, there have been a lot of efforts to simplify the setup and control of experiments over this distributed testbed. These efforts focused mainly on providing a graphical user interface to select nodes, execute commands, and perform different measurement tests. These tools typically take advantage of the PlanetLab Central (PLC) which provides detailed information about PlanetLab nodes such as host name, geographical location, TCP connectivity. It also provides convenient interfaces for adding/removing nodes. Planet lab manager [13] is one of these tools. It basically allows users to choose nodes for their slices and then to choose some of these nodes for an experiment based on the common information about these nodes. It has the capability of deploying experiment files and execution single/multiple command on every node in parallel. It also provides means to monitor the progress of an experiment and to view the output from the nodes.

PlanetLab application manager [14] is another tool that was designed to simplify deployment and monitoring and running applications over PlanetLab. It has several features that enable users to centrally manage their experiments and monitor their applications. The tool is composed of two components: the server side which requires access to aPostagreSQl/MySQL Database element and a *PhP* web server to allow web access. On the other hand, the client side is basically a shell script that runs under bash. The client side shell scripts require specific customizations making it a little bit complicated.

Stork is a software installation utility service that installs and maintains software for other services [15]. It allows users to install software on a large number of PlanetLab nodes and keep that software updated for a long period of time. It also has and a novel security system that empowers users and package creators while reducing trust in a repository administrator. As a results Stork provides a secure, scalable, effective way to maintain tarball and RPM packages and software on large networks.

Other tools have been developed to simplify the process of evaluating the characteristics of PlanetLab nodes, thus allowing users to select suitable nodes for their experiments. e. g., pShell [16], CoDeeN [17] [18]. The pShell is a Linux shell that provides basic commands to interact with PlanetLab nodes. However, for pShell users, they still need to manually select suitable nodes.

ConTest share many of these features with other tools. The most obvious difference between ConTest and other tools is that ConTest provide a visual way to select nodes within a slice using the world map view. It also, allow users to visually construct the network topology and project it on the map. Moreover, users can specify the role of each node to be client, server, or forwarding node.

VI. Conclusion and Future Work

In this paper, we have discussed the design and implementation of ConTest, a graphical user interface-based tool that enables researchers to visualize experiments over PlanetLab network. ConTest is a powerful graphical tool that helps PlanetLab users access nodes within their slices, upload files and execute commands easily, with no complications and with much fewer security restrictions compared to the command line access, which is the default way of accessing PlanetLab network. It also helps users perform different tests on the nodes they choose and automates gathering and summarizing the results for these tests. Also, it gives the users the ability to utilize its connection automation to run their own tests using their custom testing applications. This tool would be of a great help So this tool is a very promising tool as it has unlimited improvement possibilities which we hope we will be able to explore to make it a more effective tool in the world of distributed applications testing and development.

As part of our ongoing efforts to improve ConTest's functionality and usability, we seek to achieve the following goals:

- In the current implementation, the user can browse remote machine folders using the browsing commands (like ls). Future release of ConTest will enable browsing directories on a remote node graphically to increase the usability and ease of use.
- Enhance our tool's testing capabilities by adding some additional statistics. Users of the tool can easily help in improving this aspect since the test codes are separated from the main tool functionality, so virtually any codes can be tested and used.
- Improving the graphical design of the tool. Specifically the map view. In particular, we will increase the accuracy of the coordinate mapping and add the zooming feature to make the map interface easier to use.
- Even though we enable the user to choose the test codes, it is still, somehow, a limited feature. On the far view, we want to enable the users to conduct their own experiments based on the connection setup they choose, instead of limiting them by simple connection testing codes.

© 2012 ACADEMY PUBLISHER

ACKNOWLEDGMENT

The authors would like to thank Dr. Mohammad Fraiwan for his constructive thoughts and comments to improve this work.

REFERENCES

[1] PlanetLab. [Online] http://www.planetlab.org.
[2] [18] M. J. Freedman, E. Freudenthal, & D. Mazires, (2004) "Democratizing content publication with coral", In *Proceedings of the 1st USENIX Symposium on Networked Systems Design and Implementation (NSDI '04)*.
[3] V. Ramasubramanian, & E.G. Sirer, (2004) "O(1) lookup performance for power-law query distributions in peer-to-peer overlays", *In NSDI* , pp 99-112.
[4] K. Park, & V.S. Pai, (2006) "Scale and performance in the coblitz large-file distribution service", *In Proc. 3rd Symposium on Networked Systems Design and Implementation (NSDI 06)*.
[5] N. Spring, D. Wetherall, & T. Anderson, (2002) "Scriptroute: A public internet measurement facility".
[6] Nokia Corporation, Qt Creator, qt.nokia.com/products/developer-tools, 2010.
[7] Qt Centre Forum - Qt Centre Community Portal, www.qtcentre.org.
[8] Charles Morin, Randy, "How to PING", 2001.
[9] Forouzan and Behrouz, "Data Communications And Networking", 2007.
[10] Iperf project, sourceforge.net/projects/iperf, 2008.
[11] Holz, T., Gorecki, C., Rieck, K., Freiling, F.: Measuring and detecting fast-flux service networks. In: Proceedings of the Network & Distributed System Security Symposium (2008).
[12] Passerini, E., Paleari, R., Martignoni, L., Bruschi, D.: FluxOR: detecting and monitoring fast-flux service networks. Detection of Intrusions and Malware, and Vulnerability Assessment pp. 186–206 (2008).
[13] PlanetLab Experiment Manager, http://www.cs.washington.edu/research/networking/cplane/
[14] R. Huebsch. Planetlab application manager. http://appmanager.berkeley.intel-research.net/, 2004.
[15] J. Cappos and J. Hartman. Why it is hard to build a long-running service on PlanetLab. In Proc. Workshop on Real, Large Distributed Systems (WORLDS), San Francisco, CA, Dec. 2005.
[16] B. Maniymaran, pShell: An Interactive Shell for Managing Planetlab Slices, http://cgi.cs.mcgill.ca/anrl/projects/pShell/index.php/
[17] L.Wang, V. Pai and L. Peterson, The Effectiveness of Request Redirection on CDN Robustness, Proceedings of the 5th OSDI Symposium, December 2002.
[18] CoDeeN, Princeton University: http://codeen.cs.princeton.edu/

Basheer Al-Duwairi received the PhD and MS degrees in computer engineering from Iowa State University in Spring 2005 and Spring 2002, respectively. Prior to this, he received the BS degree in electrical and computer engineering from Jordan University of Science and Technology (JUST) Irbid, Jordan in 1999. He has been an assistant professor at the Department of Network Engineering and Security at JUST since fall 2009; prior to this, he was an assistant professor in the Computer Engineering Department at the same university from fall 2005 to fall 2009. His research expertise are in the areas of trusted Internet encompassing Internet Infrastructure Security focusing on DDoS, Botnets, and P2P security, Wireless Network security, and Resource Management in real-time systems. He has coauthored several research papers in these fields. He has given tutorials at reputed conferences, served as a member of technical program committee and session chair in many conferences. He also served as Assistant Dean for Student Affairs, Chairman of Computer Engineering Department, Vice Dean for Computer and Information Technology, and the Director of Computer and Information Cenetr, all at JUST. http://www.just.edu.jo/~basheer

Mohammad Marei, Malek Ireksossi, and Belal Rehani are undergraduate students at the department of computer engineering at Jordan University of Science & Technology. They are expected to graduate in summer 2011.

© 2012 ACADEMY PUBLISHER