UCLA

Easy and Flexible Experiment Management Tool

Anders Persson - anders@cs.ucla.edu
Cesar Marcondes - cesar@cs.ucla.edu

06/17/2005

Contents

1 Introduction **3**
1.1 TCP Benchmark Issues . . . 3
1.1.1 No TCP Benchmark Standard . . . 3
1.1.2 Experiment Description and Management . . . 3
1.1.3 No Global Infra-Structure for TCP Performance Tests . . . 4
1.2 Easy and Flexible Experiment Management . . . 4

2 Design Decisions **5**
2.1 Resource Server . . . 5
2.2 Control Server . . . 6
2.3 Experiment Editor - Client . . . 6
2.4 Database Front-End . . . 7
2.5 Synchronization and Security Issues . . . 8

3 Web Services **9**

4 Protocol and Information Exchange **9**
4.1 Obtaining the List of Available Resources . . . 9
4.2 Scheduling an Experiment . . . 10

5 Conclusions and Future Work **11**

List of Figures

1 TCP Benchmark Testbed at UCLA/NRL . . . 5
2 CS-RS-Editor Infrastructure . . . 6
3 Experiment Editor . . . 7
4 Resource Server Management . . . 8
5 Experiments Visualization . . . 8
6 Obtain List of Nodes . . . 10
7 Obtain Services from Individual Resource Server . . . 10
8 Scheduling an Experiment . . . 10

1 Introduction

In this section we explore briefly two related topics that motivated this work: first, the importance of doing TCP Benchmark and second, how to deal with the management of a complex TCP Experiment Testbed.

1.1 TCP Benchmark Issues

In recent years, many new advanced TCP versions have been proposed (TCP Westwood [1], FAST [2], BIC TCP [3], HighSpeed TCP [4]) to replace the widely deployed TCP Reno on the Internet hosts. The problem of NewReno is its well-known limited performance due to new and unexpected scenarios that the original Reno designers never thought before, like transport over wireless networks, satellites and over gigabit speed networks. In spite of these new TCP schemes have been supported mainly by analytical and simulation results, there is still a lack of real experiments over the Internet. These lack of real experiments can be explained by the a set of difficulties in performing and manage such distributed set of experiments over the Internet. Following are other open issues.

1.1.1 No TCP Benchmark Standard

There is no agreement on the academic community about the collection of extensive tests needed to evaluate the performance of a new TCP implementation. Such "performance criteria" and "scenarios of tests" range over a large set of possibilities, according to [5], a recent proposal on this matter in a very draft format. For example, the aggregate goodput (capturing the behavior of multiple TCP flows), fairness (max-min fairness), potential delay and backlog at bottleneck routers are among the possible candidates to evaluate 'Equilibrium Properties' as performance criteria even though some controversial exist on the fairness evaluation.

On the other hand, [5] also describes 17 possible tests suites, like congestion collapse generation, scalability tests, responsiveness tests among others, that must be performed in a controlled environment. Although such controlled experiments are necessary for TCP design phase, one can still argue that TCP transport duty has to work on real networks where the link matrix loads are not know in advance. Ultimately, these extensive tests have to be repeated on the Internet itself, without proper knowledge of network conditions, in the process of a protocol evaluation as it happens in a regular Database Performance Benchmark, using a live system.

1.1.2 Experiment Description and Management

Usually, it's very time consuming for the researchers to set up the machines, process the data and coordinate human and machine resources to perform experiments. Performance analysis of such distributed system requires automatic coordination and the awareness of the remote partners that some tests could impact their networks.

Suppose, one remote partner allows a tester to use a traffic generator to send non-responsive traffic like UDP at high rate, definitively such traffic could affect the normal traffic flowing through this site. It's part of the responsability of the tester to not allow that to happen, but sometimes misconfiguration may still exist when one is relying on test scripts, and there is no way to avoid such type of problems. Therefore, one architecture that allows, one to use the available and remote-party constrained resources and applications is necessary.

1.1.3 No Global Infra-Structure for TCP Performance Tests

Finally, the third issue and motivation is that "Global Internet infra-structures" such as PlanetLab [6] were designed to test functionality of new pervarsive applications across the Internet, not to test real performance through instrumentation over the Internet. This can be best explained because the machines that form such global pool are shared by many researchers using "virtual slices of machines (virtual machines)" such that their experiments can run in parallel, potentially affecting a serious study of transport performance such as the TCP one in this infra-structure. Hence, it's interesting to test on a new approach that removes the possibility of parallel tests and present tests that can run only on a all-or-nothing basis. This way, the regular Internet traffic is the only background traffic that we are inferring to or testing with.

Another issue is that as the Internet link speed grows up, the machine internals performance (hardware and OS) become the bottleneck for high speed transport like TCP. In such scenario, reusing the machine for multiple concurrent users applications can increase the possibility of insert "machine noise" and further impact performance results.

1.2 Easy and Flexible Experiment Management

This project proposes the development of a prototype infra-structure that will ease the burden of performing TCP measurements in a large scale Internet testbed. It turns out, however that the project has much more flexibity than that, it also provides a easy way to manage and guarantee remote party constraints in distributed experiments.

It's relevant to point out that in the past, we had performed TCP experiments over the Internet [7] [8] using a testbed that involved many partners and research collaboration (as can be seen in Figure 1). So, we have experienced how "time comsuming" is to setup experiments and manage resources across boundaries. Therefore we understand that this project will bring many beneficts to end-users like: 1) Reduce the amount of time needed to create experiments. Since it does provide an interface that gives the user a centralized view of the available resources that he can use to coordinate experiments; 2) Simplify the management of resources by allowing network administrators to control what resources should be available and how they can

be used; 3) Automate the otherwise tedious task of data collection, and extraction; 4) Finally, it provides storage of the data which can then be posteriorly queried and analyzed.

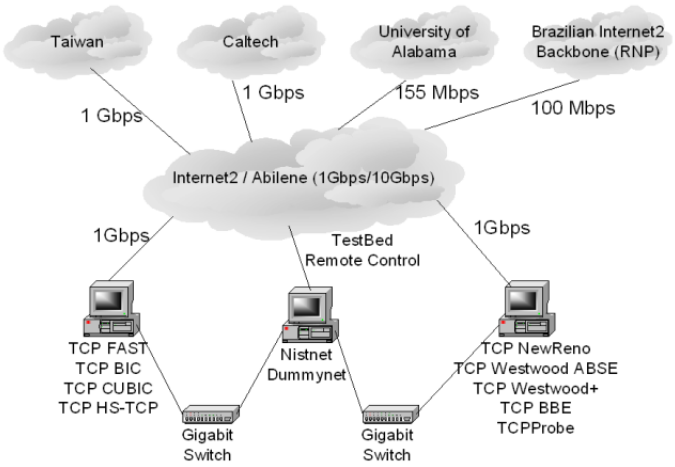

Figure 1: TCP Benchmark Testbed at UCLA/NRL

2 Design Decisions

In this section we explain, in detail, the developed *major* components of the prototype system (Resource Server, Control Server, Experiment Editor and Front-End to Database) that we created, and describe the assumptions and decisions made during the design process.

2.1 Resource Server

The Resource Server has a central role in our archicteture. One important assumption that we made is: **DO NOT** assume anything about the remote machine. It does not require any particular operating system (the Resource Server software was written in Java), it does not require any type of synchronization (the Control Server provides such synchronization by controlling the scheduling and execution), it does not require to keep permanently the data files of experiments. All control commands are sent through SSL so it's not necessary leave any well-know unsecure port open in a firewall.

Through the Experiment Editor, the Resource Server provides the system with a full view of the available resources on a particular host. Such description of the services are very flexible and easy to understand,

and it also dictates "which" system services will be available and over what type of constraints put on those resources (minimum, maximum, required option) depending on the remote party policies.

Once the Resource Server receives a task (an experiment) from the Control Server scheduler, it executes the applications related to the services and generates corresponding output files temporarily. If the task configuration has a specification that lies out of range, the Resource Server denies access to its resources and returns the correspondent code of what constraint was violated. Otherwise, it declares success and start the scheduled experiment at the correct moment.

2.2 Control Server

The Control Server is the hub, brain and database interface of the system. It provides available resource information to the Experiment Editor (client) such that Experiment Designer does not need to know at any moment the real addresses of the remote machines. For this kind of communication, the Control Server acts as a relay. However, the real importance of the Control Server consists on the scheduling mechanism from experiment descriptions provided by the clients, coordination of the resource server execution and the management of the experiments database. The last comment about the Control Server is that whenever an experiment finishes the Control Server is responsible to collect the log files and store it into the database. We implemented the Control Server in Java and the database is an open-source mysql.

The following figure (Fig. 2) presents the connectivity of Experiment Editor and available Resource Servers through the Control Server.

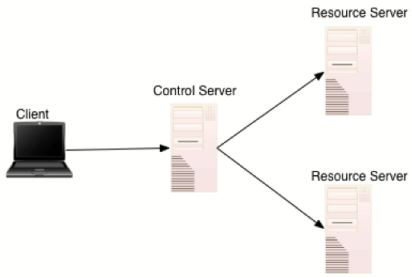

Figure 2: CS-RS-Editor Infrastructure

2.3 Experiment Editor - Client

The Experiment Editor is responsible for presenting the information that is provided by the Control Server in a way that is easily understood by the user, by a proper visual interface. Once the Editor start a new Ex-

periment, it obtains a list of "available" resources so that the task of create an experimental outline consists only in choosing the Resource Servers that are going to be used, connect them and submit the final result to the Control Server for scheduling.

The Experiment Editor starts totally clean and does not have any information about the topology or what resources are available. As messages are exchanged with the Control Server, the central panel builds up automatically depending on the content of these messages and then it's possible to prepare an interaction graph of the Resource Servers components that are going to participate in an Experiment.

For every Resource Server chosen to participate, a correspondent floating button appears in the central panel of the Editor and there is the possibility of connect those with lines (and semantically connecting services too), like a regular diagram drawing tool. The next figure (Fig.3) shows one screenshot of Experiment Editor with "3" Resource Servers connected and one of the Services configured. This software was developed in Java with the graphical library Swing [9].

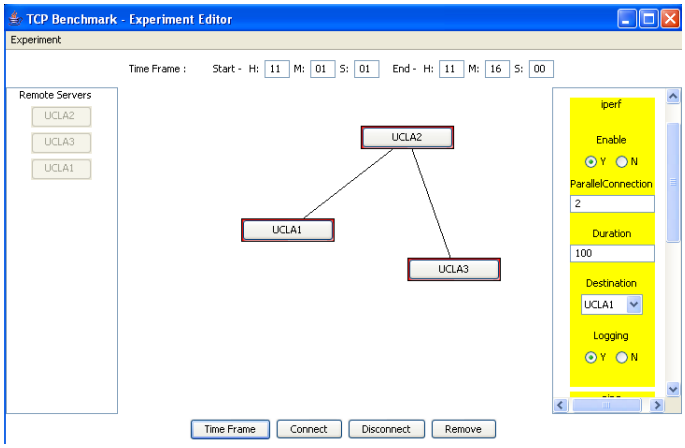

Figure 3: Experiment Editor

2.4 Database Front-End

The final piece of our Experiment Testbed Infra-Structure is the Database Front-End, it consist of a website in HTML and java servlets accessing continuously the Mysql database adding, deleting, searching and presenting information about Resource Servers on this system, as well as Experiments that are been executed. If the experiment has already finished its execution completely, the Front-End exhibits the DataSet related to a specific experiment such that the experiment designer can verify the resulting files of executing the scheduled experiment.

Possible files to be visualized are trace files of common application like traceroute, iperf, ping, tcpdump, capprobe, spruce estimations, among others.

The following 2 figures present in detail the interface. In Fig.4, we have a list of the Resource Servers registered with the Control Server, the addition of a new Resource Server is as easy as just fill the form. The Fig. 5 presents some Test Experiments performed during the demo presentation of CS217.

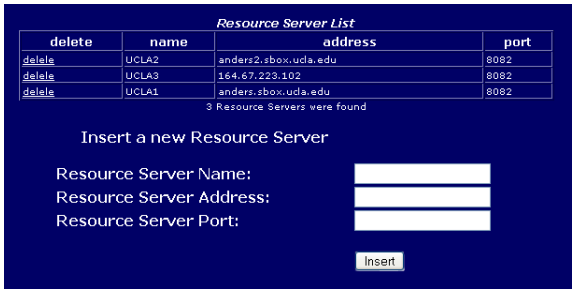

Figure 4: Resource Server Management

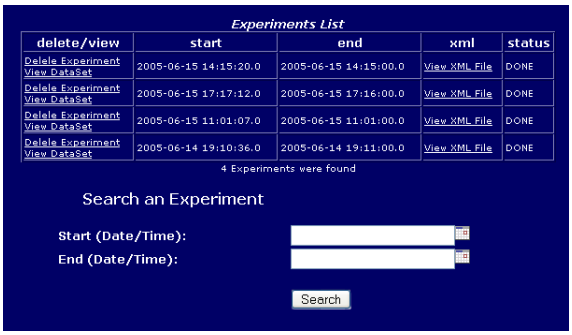

Figure 5: Experiments Visualization

2.5 Synchronization and Security Issues

Some special issues like synchronization and security were also designed such as the main idea of "no assumptions" under the Resource Servers could still hold true. For the case of synchronization, all the scheduling and commands of execution are propagated from the Control Server at appropriate times, therefore only the Control Server needs to have a good clock precision. All the measurements outputs are defined to be relative to that start time of the experiment. However, such design aspect has a drawback, if the Resource Server machines are in different networks under different Round Trip Times, it is necessary that the Control Server sends in advance (with a proper delay) the execution message depending on an estimated value of the Control Server and Resource Server. Since, precision in start times is not a big issue on TCP experiments such heuristic works well in this system.

In term of security, we developed the tools to support SSL (Socket Security Layer) that relies on (self-signed) certificates to cypher the communication. This is quite useful when sending commands to the Control Server, since there is no proper authentication phase, and the Experiment Editor can be known by the certificate its using. However, the collection of data, the last part of the streamlined execution process, has to be executed using non-secure methods and even multiple FTP connections since such transfer of big dump files can become really slow whenever using cryptography to cypher the raw data.

3 Web Services

In this project, we used a fairly amount of XML in the description of resources, experiments, constraints. XML has great deal on the flexibility of our system since it has meta-data as the basis of the message formats. Therefore, all the messages are self-described. When the Experiment Editor receives a message describing the resources, by the meta-data contained in the XML service file, it can figure out what kind of fields are necessary to draw. Additionaly, it provides extensibility since new services can be rapidly created even when the remote application is new or has different arguments. Since it's up to Resource Server manager to especify the services available, the options allowed and the type of fields, such XML-based messaging is appropriate and ease to manage.

We can relate our work with the most recent "Web Services technology" that are emerging as standards for distributed internet computing [9]. Web Services uses the same XML-based protocols and create frameworks that are independent of a particular language or programming model what fits exactly with our conceptual model.

4 Protocol and Information Exchange

In this section, we describe our internal protocol and the message flow of the system, from the list of available resources to the final execution of an experiment. The protocol resembles HTTP since it's based on Request/Response messages and futhermore since the payloads are XML-based it also inherit some characteristics of the SOAP [10] Web Services protocol.

4.1 Obtaining the List of Available Resources

Initially, the Experiment Editor starts without information about the available resources, it sends to a preconfigured and certification signed Control Server, a TOPOLOGY message. The Control Server sends back a list of Resource Servers in XML format. However, it does not send the available services related to those Resource Servers, since this initial message exchange could grow very fast since there could be a number of possible services at each of the Resource Server (Figure 6).

Once the experiment designer chooses the Resource Server he wants to work with, in an experiment. Then, for every Resource Server choice, there is a message exchange between Editor and Control Server, so that it can obtain the specific list of "services" that weren't brought in the first place. It is important to point out that the Control Server does not keep this services list in the Database, so it must contact the respective Resource Server and ask in behalf of the Experiment Editor for the available services, NODEINFO protocol message. Such decisions represents the intention of 1) loads off the responsability of the Resource Server to keep updated the Control Server with its services. 2) The Experiment Editor does not need to know the

certificate keys of the Resource Servers, all the communication can be relayed by the Control Server (Figure 7).

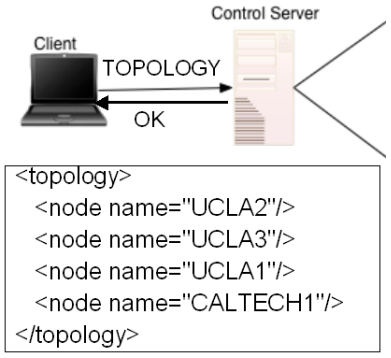

Figure 6: Obtain List of Nodes

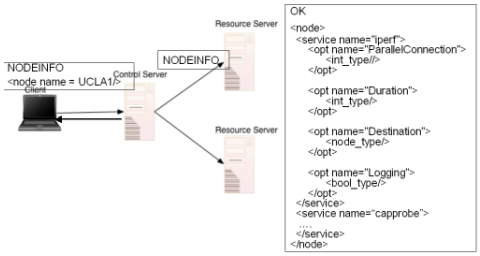

Figure 7: Obtain Services from Individual Resource Server

4.2 Scheduling an Experiment

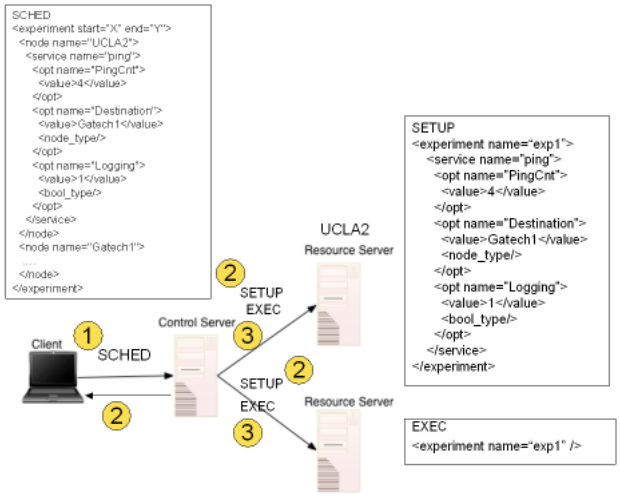

Figure 8: Scheduling an Experiment

Once all the services are available to the Experiment Editor, the designer can model an experiment based on what he can see in the main window (fig.3, page 6). The Editor allows the designer to prepare an interaction graph of the resources so that traffic generator tools, as an example, can generate traffic that

flows between any two nodes. After this interaction graph is done, it's up to the designer to schedule the time when this experiment will be run. The client then sends the complete experiment to the Control Server using the protocol message SCHED and the payload experiment XML. At this point, the Control Server has some responsabilities (Figure 8):
1) Check the validity, in terms of scheduling, of this experiment, it cannot allow time overlapping experiments since the system is based on a all-or-nothing philosophy.
2) It strips off the experiment XML, such as mini-messages of SETUP can be sent to each of the related Resource Servers involved.
3) If no violation of constraints happens, and all Resource Servers reply successfully to the Control Server, and only then the experiment is said to be "valid". The Control Server then creates one entry of the experiment in the database, send OK message back to the client, and at appropriate delayed times send the specific execution messages to the Resource Servers referring the previous setup.

5 Conclusions and Future Work

This project was basically a prototype software development and a proof of concept. Additionally, in this document, we discussed at first the importance of establishing a standard framework for TCP and also we motivated by the need of an easy and flexible way to manage a distributed experiment testbed, where such TCP experiments can be executed smoothly. One strong assumption was that the remote machines provided by collaborators do not need to have any special software/hardware functionality, and the configuration of administrative boundaries and parameter constraints was built-in the system using XML. The system proved to be easy to manage, in the demo presentation, we showed 1) how to create an experiment involving 3 machines, 2) how the constraint violations work, 3) how to setup a new Resource Server machine, 4) Execute a full experiment and collect the related output data, all these in about 8 minutes.

We intend, as future work, explore the power of Web Services in order to create a way to filter out data, since the output of any application is quite specific. Furthermore, the data, we are interested in (tcp kernel traces), is usually formatted differently depending on the machine and operating system, it should be up to the Resource Server to specify how to filter such data, and once the filter is done give meaning to the data. This way, the visualization can be done in a more intelligent way, where the Front-End can embbed a graphical tool like Gnuplot, where the designer can select based on the self-described fields which data to visualize.

Another possibility is augument the Experimentor Editor with Artificial Intelligence such that once it obtains a list of resources, it can prepare automatically the types of tests needed for a complete TCP Benchmark in the Internet. For example, by scanning the Internet topology using remote estimation tools like CapProbe [11], Spruce [12], tcptrace and so on. Such intelligent application can verify which tests can fit in an appro-

priate moment.

Finally, we intend to verify the possibility of integrate such flexible system with PlanetLab, even though it has different design decisions but the configuration management is something useful for any Global Internet testbed, or with Emulab [13].

The source code of this project is available at http://anders2.sbox.ucla.edu:8080/benchmark/.

References

[1] M. Gerla, M. Sanadidi, B. K. F. Ng, M. Valla, and R. Wang, "Tcp westwood with adaptive bandwidth estimation to improve efficiency/friendliness tradeoffs," *Journal of Computer Communications, Volume 27, Number 1, January 2004*, 2004.

[2] C. Jin, D. X. Wei, and S. H. Low, "Fast tcp: motivation, architecture, algorithms, performance," *IEEE INFOCOM 2004 Conference on Computer Communications*, 2004.

[3] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," *IEEE INFOCOM 2004 Conference on Computer Communications*, 2004.

[4] S. Floyd, "Highspeed tcp for large congestion windows," *IETF RFC 3649*, 2004.

[5] D. X. Wei, "Draft version - available at http://www.cs.caltech.edu/ hegdesan/sigcomtest/," *Benchmark for TCP Congestion Control*, 2004.

[6] A. Bavier, M. Bowman, B. Chun, D. Culler, S. Karlin, S. Muir, L. Peterson, T. Roscoe, T. Spalink, and M. Wawrzoniak, "Operating system support for planetary-scale network services," *First Symposium on Networked Systems Design and Implementation*, 2004.

[7] C. Marcondes, A. Persson, M. Sanadidi, and M. Gerla, "Tcp westwood experiments over large bandwidth delay product networks," *International Workshop on Provisioning And Transport for Hybrid Networks (in conjunction with IEEE Broadnets 04)*, 2004.

[8] C. Marcondes, A. Persson, L.-J. Chen, M. Y. Sanadidi, and M. Gerla, "Tcp probe: A tcp with built-in path capacity estimation," *The 8th IEEE Global Internet Symposium*, 2005.

[9] K. Walrath, M. Campione, A. Huml, and S. Zakhour, "The jfc swing tutorial: A guide to constructing guis, second edition," *Addison-Wesley Professional*, 2004.

[10] W3C, "Simple object access protocol (soap) 1.1," *Note 8, 2000 - http://www.w3c.org*, 2000.

[11] R. Kapoor, L.-J. Chen, L. Lao, M. Gerla, and M. Y. Sanadidi, "Acm sigcomm '04," *CapProbe: A Simple and Accurate Capacity Estimation Technique*, 2004.

[12] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," *Proceedings of the SIGCOMM Internet Measurement Conference '03*, 2003.

[13] S. Guruprasad, L. Stoller, M. Hibler, and J. Lepreau, "Scaling network emulation with multiplexed virtual resources," *SIGCOMM 2003 Poster Abstract*, 2003.