Methodology definition for reliable network experimentation

René Serral-Gracià, Xavier Masip-Bruin, Marcelo Yannuzzi, Eva Marin-Tordera
Advanced Network Architectures Lab (CRAAX), Barcelona TECH (UPC), Spain
{rserral, xmasip, yannuzzi, eva}@ac.upc.edu

***Abstract*—As researchers in the networking area keep adopting experimental network testing as a valid mechanism to develop, validate, and improve their research, it becomes more apparent that an overall framework supporting and assisting during the experimentation process is necessary. Particularly, this assistance is relevant in processes such as experiment preparation, or results validation. As a consequence, the goal, and thus the contribution, of this paper is twofold, on the one hand we propose a novel set of guidelines which establish the set of requirements any testbed for network experimentation should follow. On the other hand, as the other relevant contribution of this work, we propose a mechanism for generating meta-data information on the experiments that ease the publication of the obtained datasets. Finally, as a use-case, we present a particular implementation of this framework which we deploy in a real scenario to prove the capabilities of the proposed testing procedure.**

I. Introduction

Experimental research is being adopted as a valid alternative to analytical and theoretical approaches. Nevertheless, when experimenting in networking environments it becomes apparent that a very well devised methodology is required to provide a solid testing procedure and correct results interpretation.

Aligned to that, when performing experiments, researchers often wonder whether the results are correct or some *"under-the-hood"* unexpected behavior is crawling into the obtained results. In this regard, having guaranties about the outcome of the experiment is desirable. Moreover, the issue is exhacerbated by the lack of a reference source, which makes the assessment of the experiment outcome more difficult to achieve.

To alleviate these concerns, in this paper we present both a well-defined methodology and its practical implementation of a Testbed as a Service (TaaS). Briefly, our proposal provides a set of guidelines and best-practices used to ensure that the whole testing procedure is correctly performed and that the obtained results minimize any existing bias. Then, in order to validate this proposal we present, as a use case, a particular implementation of such approach in a real deployment, were we perform a series of QoE assessment experiments of multimedia video streaming, to later publish them as reference dataset for future researchers. Such dataset contains self-generated metadata that can be used in order to understand and provide mechanisms in order to repeat such experiments in different environments.

This work was partially funded by Spanish Ministry of Science and Innovation under contract TEC2009-07041, and the Catalan Government under contract 2009 SGR1508.

The rest of the paper is organized as follows, in the next section we present some related work both in the area of testbeds, and public data sets, particularly in QoE assessment. Afterward, in Section III we propose our TaaS framework, we continue in Section IV with the description of the use case of our particular implementation, namely TManT, and the mechanisms used in order to publish the obtained results. Finally, in Section V, we conclude and draw the lines for our future research in this area.

II. Related Work

Nowadays experimental research is a trendy topic in many areas. Consequently, more and more public testbeds appear in order to facilitate this testing. Examples of such testbeds range from Living Labs such as Botnia Living Labs [1], to the pure network testing available in testbeds such as PlanetLAB [2], or most recently in Emulab [3] and others. Nevertheless, all these testbeds share a common treat, that is, lack of specific capabilities for assisting the experimentation, nor any validation features on the obtained results. Opposed to this, the framework we are proposing in this paper, allows the experimenter to be constantly aware about the most common issues presented during the network experimentation, e.g., timestamping accuracy, resource overloading, unintended network behavior, and so on. Added to this, our proposal also allows to conditionally and automatically repeat any experiment not complying with some previously specified criteria.

In the subject of testbed federation, new proposal, namely, *cOntrol and Management Framework* (OMF) [4] is starting to develop some momentum, as it allows seamless experiment control and execution between federated testbeds, currently supporting PlanetLAB and NITOS wireless testbed. The main difference of such initiatives and our proposal is that our framework is focused on network experimentation, and thus offers exclusive features to deal with this scenario.

Another focus of our work resides on the seamless publishing of the obtained experiments, specially through the integration of meta-data describing the testing process. In this regard, there are also other efforts that aim at this dataset publication. In the particular area of multimedia Quality of Experience assessment, and video encoding, there are approaches such as LiveVQ-Database [5] that offer videos with added synthetic disruptions and impairments, plus different compression codec that permits to compare the differences among them. Other approaches such as [6] are focused on the analysis of 3D stereoscopic videos. The main difference our proposal offers

is that it is network oriented, and thus the results which might be potentially extracted involve the testing in a real environment, and are intended for network researchers. Moreover, the novel meta-data extraction mechanism we devise has not been proposed by any other initiative as far as we know.

III. Testbed as a Service

As we have already shown, the diversity of testbeds and their different scopes makes it very complex to test network technologies, or distributed applications in a reliable way. In this paper, we propose a *Testbed as a Service* (TaaS) framework, where the main focus relies on presenting a testbed where the experimenter has full control over the network configuration, being able to create, delete, or change the properties of any link in the topology, or seamlessly generate traffic among any node in the network. The testbed allows easy access to these operations by populating all the testbed features to third parties as a set of different services, easy to access both from the management interface, and from the different APIs populated.

In this section we present the prerequisites and our particular definition of such a framework, together with its main capabilities that guide the experimenter all over the testing procedure to provide a solid and reliable platform. Then, we also describe how this framework may be used in order to create public datasets, particularly for QoE assessment.

A. Testbed requirements

When designing a new testbed it is important to consider several aspects of the testing procedure to provide a reliable and accurate platform. According to [7], measuring network performance must follow some best-practices that ensure proper measurements. Particularly: *i)* we have to deal with errors and imperfections in the measurement process, i.e., precision, accuracy, and equipment calibration; *ii)* the performed experiments must be repeatable to determine their validity, *ii)* meta-data of the experiment must be gathered and made available to help third parties at understanding the mechanisms used to perform the experiment, and finally to provide assistance in the publication of the obtained results.

Then, to cope with these best-practices we provide a testbed with the following features, *i)* Control over the resources, and the overall system, in order to guarantee that the resource usage does not bias the results; *ii)* Extensible, to provide means of upgrading the facilities without disrupting the testbed itself; *iii)* Flexible, generally different tests might require radically different setups that the testbed should be able to exchange easily.

B. TaaS building blocks

The TaaS proposed in this work has the objective of assisting during the different phases of an experiment, from the testbed preparation and setup, to the results gathering. We achieve this by providing a six steps method that guides the experimenter through the whole testing procedure. In Fig. 1 we summarize the different necessary steps.

As it can be noted in the figure, the platform is composed by three different parts, *i)* the Testbed Management Module (TMM), *ii)* the Test Orchestration Module (TOM), and *iii)* the Management Interface (MI).

TMM is responsible for creating the particular network topology, setting up the links, and scheduling the different events for the tests. TOM is in charge of executing the necessary actions to run the experiments, while finally MI acts as frontend of the testbed with the experimenter. A detailed description of these modules is introduced next.

1) Testbed Management Module: One of the key parts in experimental research is the existence of a testbed where the experiments can be performed. However, tasks such as managing, maintaining, extending, or configuring it are often slow, tedious, time consuming, and since they are human assisted, they tend to be error prone and hard to validate.

In our proposal, the Testbed Management Module (TMM) is responsible for controlling the testbed. More specifically, it performs all the required tasks in order to properly configure and prepare the testbed for the experiments. This configuration is divided into three different subsystems, *i)* Virtual Machine Subsystem (VMS), *ii)* Network Configuration Subsystem (NCS), and *iii)* Network Events Subsystem (NES).

a) Virtual Machine Subsystem (VMS): VMS is the subsystem in charge of creating the default configuration for the different nodes composing the experiment. Thus, depending on the experiment demands, the VMS will decide whether it is necessary to increase the number of nodes of the testbed, and if so needed, the system would dynamically generate the necessary virtual machines with the specific requirements demanded by the experiments, and later will deploy such virtual machines into the assigned physical machines.

This permits to scale the testbed up to larger network dimensions with sustainable hardware requirements, providing *extensibility* in the testbed.

VMS expects as *Input* a set of physical machines interconnected through a network, and returns as *Output* a set of Virtual and Physical Machines complying with the requirements of the experiment in terms of CPU, RAM, Disk Space, Network Interfaces, and Operating System.

The created virtual machines will be cloned from one or several masters, which will have all the required applications and configuration to execute the experiment.

b) Network Configuration Subsystem (NCS): When all the nodes—both virtual and physical—are deployed, then the NCS is in charge of configuring all the network equipment. This is accomplished by setting the proper addresses in the interfaces, along with the configuration of the necessary VLANs in the switching infrastructure, the routes in the routing equipment, and if necessary the networking protocols—e.g., routing, management, and so on.

NCS expects as *Input* a set of configured nodes, a topology description, and the network configuration. The nodes are obtained from the VMS, while the topology and the network configuration are provided by the user. NCS returns as *Output* a set of interconnected nodes complying with the particular topology and the set of required protocols, running with the assigned configuration.

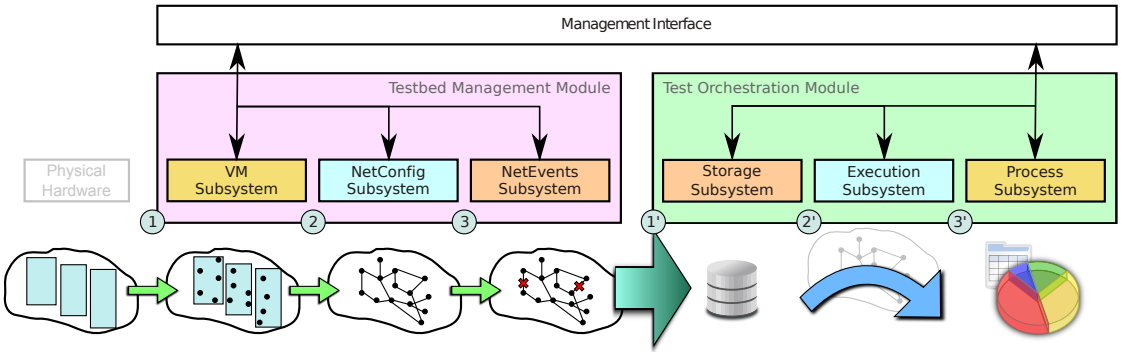

Fig. 1. Testbed design and internal architecture

c) Network Events Subsystem (NES): The Network Events Subsystem (NES) deploys the initial network disruption scenario—e.g., down links, bandwidth limits, faulty links, inter-continental links, congested links, background traffic, and so on—and schedules the different changes in these conditions that will occur during the experiment.

NES expects as *Input* a configured testbed with a network topology, an initial network status, and a list of events to trigger during the experiment. As *Output* it returns a configured testbed with a set of scheduled network events ready for the experiment to start.

NES is the last subsystem in the TMM pipeline before passing the control to the Test Orchestration Module, which will perform all the tasks related to the experiment execution and results extraction.

2) Test Orchestration Module (TOM): Once the testbed has been tuned and configured, the TOM will setup the storage subsystem, execute the experiments, and finally will process and will deliver them to the user following a specified criteria defined in advance.

Similarly to the TMM, TOM subsystems are executed sequentially. But given the different requirements of the experiments, the system can execute the three subsystems concurrently. The decision whether to run the tasks sequentially or concurrently is taken by the experimenter.

a) Storage Subsystem (STS): The Storage Subsystem sets up the required raw results gathering operations, e.g., by running traffic capturing processes, or by running daemons listening to certain protocols, depending on the particular experiment. It is worth noticing that STS does not perform any result gathering per se, but rather sets all the different applications in order to store the results during the experiment execution. It is important to notice, that the captured information is independent of the testbed itself, the only requirement is that the experimenter sets up the applications to perform the task, while the monitoring and tracing of their execution is performed by the internal testbed features. As a consequence, the data gathering can be performed at multiple layers at the same time, from network level traffic collection, to video streaming dumping to disk.

STS expects as *Input* a configured testbed, and the location—or locations—where to store the results, and returns as *Output* a configured storage system ready to gather the required information by the experiment.

b) Execution Subsystem (EXS): The Execution Subsystem has two modes of operation, *i)* sequentially running each subsystem, where EXS actually executes the experiment, and *ii)* running several subsystems concurrently, where the test execution is deferred until the Process Subsystem has finished its operations.

EXS expects as *Input* a configured testbed and an experiment to execute, and returns as *Output* a finished experiment and raw experiment results in sequential mode, or an scheduled experiment and a ready storage system if operating concurrently.

c) Process Subsystem (PRS): The Process Subsystem is in charge of processing the obtained raw results by the running experiment and generating the output expected by the user. Analogously to the EXS, it has two different modes of operation, when running sequentially it runs once the experiment is finished to gather the expected results, otherwise, it runs concurrently to the experiment.

PRS expects as *Input* the raw experiment results and a finished experiment when working in sequential mode, or an scheduled experiment with a ready storage system if running concurrently. While its *Output* is the processed experiment results ready for the user.

3) Management Interface: The management interface provides a user friendly interface between the different testbed functionalities and the end-user, aiming at, *i)* guiding during the testbed setup process; *ii)* allowing the execution of the configured experiments; *iii)* extracting the results, and finally; *iv)* allowing results publication to be used by other experimenters.

The most relevant feature present in the Management Interface is the meta-data description of the experiments and the obtained results. This is achieved using the following procedure:

1) When an experimenter sets up a particular experiment, an XML file is generated describing such setup.
2) Then, this file is both used as input of the TMM, and as experiment descriptor.
3) On each step of the configuration process within the TMM, the output is another XML file with the description of all the performed operations, which will

be used by the next module as input, but also fed to the meta-data gathering process within the Management Interface.

4) Once all the steps are finished, the TOM is invoked, and similarly, each output belonging to each subsystem is used as meta-data.
5) Finally, all the information (configuration + execution) is summarized into a report that will be published together with the experiment description and results.

With the above procedure, the system itself embeds all the necessary information in order to replicate the experiment in other conditions, or more importantly to help other researchers to understand the used method, along with the capabilities of using such dataset as the base reference for other research.

IV. Use Case: QoE assessment of video streaming

In this section we introduce Testbed Management Tool (TManT), a tool implementing a testbed management system based on the TaaS rationale presented so far in this paper.

The goal of TManT is to assist in the experiment preparation, equipment configuration, and later experiment execution within our controlled testbed located in the CRAAX premises [8]. The testbed is currently composed by a cluster of 10 quadcore machines plus a management station, with 8Gb or RAM memory interconnected among them through 4 Gigabit Ethernet cards per node, allowing a flexible topology configuration using VLANs. This permits to create large topologies, up to 100 nodes using virtualization, and with sufficient computational power to perform experimentation in such a system.

All the equipment is running Linux Debian with networking capabilities that allow to configure many network parameters through `netem` [9]. Moreover to assist in the routing within the network all the requested nodes can run an instance of BGP using Quagga Routing Suite.

A. Test and Testbed Setup

The executed experiment has three different goals. First, as the experiment objective, we want to assess the effects of packet losses in a network and how do they affect video flows under different conditions. Second, we want to validate the actual performance and behavior of TManT in this real-world use-case. And third, by using our platform we want to publish the obtained results as reference datasets for other researchers in the area.

To perform this initial validation we configured the testbed in a simple topology using 10 different nodes, with a topology with a total of 16 links and a maximum hop distance between the nodes of 5. We then started a video streaming session between two nodes separated 5 hops. Simultaneously, we injected background traffic between two nodes sharing links with the above, and most importantly we caused synthetic packet losses in the bottleneck link. To gather the results the system was configured to gather multi-layer information, i.e., we used `wireshark` in order to acquire network level information about the video stream flows at the edges of the

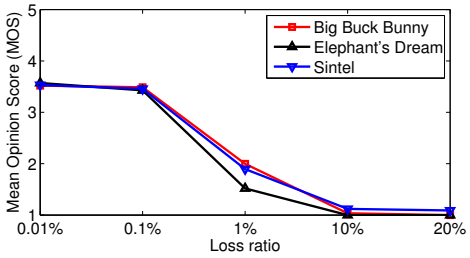

Fig. 2. Results for packet loss in 720p video with 2500Kbps bitrate

communication, while at the same time, by using `FFMpeg` we dumped the received video to disk for off-line analysis.

In order to demonstrate the test repeatability features we generated a total of 150 videos using the following combinations: we had three h.264 videos cut to 25 seconds long each, *i)* action high movement and detail video *(Elephant's Dream)*, *ii)* action high movement medium detail video *(Sintel)*, and *iii)* low movement high detail video *(Big Buck Bunny)*. Each one of them using all the combinations with a packet loss ratio of 0.01%, 0.1%, 1%, 10%, and 20%, resolutions of 720p, 480p, 360p, and 240p, and bitrates ranging from 1750Kbps to 512Kbps.

B. QoE assessment

Since the focus of this paper does not reside in a deep analysis of video quality assessment, but rather on the evaluation of TManT as a platform, we limit our analysis to the main outcome of the experiments, without entering on the internal details and methods used in order to obtain them.

In the Fig. 2 we detail the effects in the video disruption in different packet loss ratios. As it can be observed, the perceived quality of the videos drops considerably—from good to very bad quality—depending on the amount of disruptions encountered in the network.

Another interesting observation, is that the video more sensitive to packet losses is Elephant's Dream, the cause for this behavior is that with action movies showing many details during action scenes, the loss of packets is more noticeable than in other environments. In Fig. 3 we visually compare a particular frame with different packet loss ratios. Particularly, we can observe the reference frame, jointly with the 0.1, 1, and 5% packet loss ratio. Accordingly, the perceived quality is reduced.

C. Platform validation

To validate TManT, we enabled experiment monitoring to get the resource usage and the accuracy in the application of the network disruptions. Particularly, this was achieved by actively monitoring the CPU status of the involved machines, and at the end of each we, in an off-line validation process, computed the effectively observed packet loss ratio, comparing its value with the intended objective. The resulting values for all the tests can be observed in Table I. As it can be noted, there is a minor deviation between the intended and the observed packet losses. Despite of this, the deviation is marginal, and as statistically expected, is more noticeable

(a) Source (b) 0.1 Loss (c) 1 Loss (d) 5 Loss

Fig. 3. Image quality comparison

for very low loss ratios, where the discrete nature of the losses makes it very complex to reach the expected value—as determined by the experiment—with low rate flows, such as regular video streaming.

Exp. loss	Mean	Std. Dev.	Avg. Repetitions
0.01	$5.86e-03$	$9.07e-03$	2.1
0.1	0.08	0.04	1.5
1	0.9	0.1	1.2
5	4.9	0.2	1.01
10	9.8	0.5	1
20	19.9	0.5	1

TABLE I. EXPECTED LOSS WITH MEAN OBTAINED LOSS, STANDARD DEVIATION, AND AVERAGE REPEATED TESTS

In order to reduce the error in inserted packet losses in the experiments, we also configured the system to automatically repeat any test with an effective packet loss ratio deviating too much from the expected value. This caused that some tests had to be repeated, in the rightmost column of Table I we show the average number of repetitions. As expected, the experiments with small loss ratios needed in average two repetitions in order to reach the expected value. This is not an issue since we use an automatic background process to detect such unintended behavior during the test execution, and thus the repetition of the experiments does not involve human intervention. Hence showing that the usage of correct methodologies for testing can provide invaluable assistance in avoiding bias or unexpected behaviors during the experimentation.

D. Meta-data extraction and publication

The last outcome of TManT is the preparation of the experiments, together with its meta-data, in order to be made publicly available for any other researcher to use them as reference for their experiments. For the purpose of this work we partially published the outcome of this research in the form of a public dataset as found in [8].

Each experiment on the dataset is composed by three different parts.

1) *Application layer information:* Composed by the resulting video files as received in the destination and the XML file describing the observed experimental QoE.
2) *Network layer information:* Composed by `tcpdump` traces both at source and at destination together with the meta-data describing how the data was acquired.
3) *Experiment meta-data:* XML file describing the testbed together with the experiment description.

V. CONCLUSION

In this paper we presented a set of guidelines and best practices that allow to reliably perform experiments in a networking environment. These guidelines offer an structured methodology to detect any potential misbehavior together with mechanisms in order to configure, prepare, execute, and finally, gather the experimental results.

As a proof of concept, we demonstrated the proper behavior of our proposal introducing TManT, our particular implementation of the TaaS framework, using it to perform testbed configuration, experiment execution, and later using the gathered meta-data to publish the experiment results. Using TManT we outlined the effects on video streaming quality caused by packet losses, together with the study of the proper behavior of TManT, when orchestrating a testbed configuration and experiment execution in a scenario composed by 10 different nodes, and different data gathering modes, i.e., network and application layer gathering. Finally, we also demonstrated how to gather meta-data information of the experiment in order to share it in a public dataset.

In this work, we left as future lines of our research the detailed study in the impact on video quality caused by network disruptions. Particularly, packet losses, but also high network delays, delay variations, and corrupted packets. In regard of our TaaS framework, we plan to extend our evaluation to larger network topologies, using more complex components in order to further evaluate the advantages of using such a system.

REFERENCES

[1] A. Sälstorm, "Botnia living labs," 2012. [Online]. Available: http://www.openlivinglabs.eu/node/125

[2] P. University, "Planetlab," 2012. [Online]. Available: http://www.planet-lab.org

[3] "Emulab," 2012. [Online]. Available: http://www.planet-lab.org

[4] T. Rakotoarivelo, G. Jourjon, M. Ott, and I. Seskar, "Omf: a control and management framework for networking testbeds," *Operating Systems Review*, vol. 43, no. 4, p. 54, 2009.

[5] K. Seshadrinathan, R. Soundararajan, A. Bovik, and L. Cormack, "Study of subjective and objective quality assessment of video," *Image Processing, IEEE Transactions on*, vol. 19, no. 6, pp. 1427–1441, 2010.

[6] L. Goldmann, F. De Simone, and T. Ebrahimi, "A comprehensive database and subjective evaluation methodology for quality of experience in stereoscopic video," in *Proc. SPIE*, vol. 7526. Citeseer, 2010, pp. 10–1117.

[7] V. Paxson, "Strategies for sound internet measurement," in *IMC '04: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement*. New York, NY, USA: ACM, 2004, pp. 263–271.

[8] R. Serral-Gracià, "Qoe and tmant dataset – craax," 2012. [Online]. Available: http://dataset.craax.upc.edu

[9] "Linux advanced routing and traffic control," 2012. [Online]. Available: http://www.lartc.org