5th ISCA/DEGA Workshop on Perceptual Quality of Systems (PQS 2016)
29-31 August 2016, Berlin, Germany

Containerisation in Multimedia Research Test Beds

Yusuf Cinar[1], *Hugh Melvin*[1], *Peter Pocta*[2], *Mohannad Al-Ahmadi*[1]

[1]College of Engineering & Informatics, National University of Ireland, Galway, Ireland
[2]Dept. of Telecommunications & Multimedia, Faculty of Electrical Engineering, University of Zilina, Slovakia

cinaryusuf@gmail.com, hugh.melvin@nuigalway.ie
peter.pocta@fel.uniza.sk, m.alahmadi1@nuigalway.ie

Abstract

This paper proposes an innovative, flexible, and easily reproducible test bed that facilitates large scale multimedia (MM) quality centric experiments. It makes use of Linux container technologies and several other utilities that collectively facilitate ease of deployment, minimal resource utilisation and better reproducibility. We outline the significant advantages of this approach over physical test beds, virtual machines and simulation and showcase it in an experiment dealing with speech quality aspects of WebRTC using the state-of-the-art quality prediction model, i.e. POLQA. However, it can be easily extended for any type of real time communication (RTC) and MM content. The particular test bed involves a single host on which multiple WebRTC based VoIP endpoints run within Linux containers. The test bed also comprises a native network emulator running between the endpoints. Several other tools are additionally incorporated in the test bed so that speech quality of the transmitted signal can be evaluated using objective quality prediction models, such as PESQ, POLQA, etc.

Index Terms: test bed, scaling, containerisation, emulator, multimedia quality assessment

1. Introduction

Real world multimedia communications, whether real time or streaming, typically take place between at least two separate machines. Test beds for research in this area often follow this practice consisting of two or more different machines with the necessary software to represent the parties of the communication and network components to connect them. As seen from Table 1, there are four main approaches to setup a test bed for research of this kind: deploying physical resources, deploying virtual resources, simulation, and container based approach. A hybrid test bed, deploying a mix of these, is also possible. It is worthwhile examining these approaches using differing metrics.

Firstly, to test scalability and different network topologies, the use of physical resources can be very expensive and complex. Relative to this, virtual test beds can realise a cheaper test bed and more flexible network topology, but can still present practical resource limitations [**?**], e.g. memory. Secondly, reproducibility can be challenging with such an implementation complexity in both physical and virtual test beds. [**?**] asserts that reproducible papers do not seem to be a standard practice in computer science related areas since reproduction necessitates access to all code and resources to produce the results. However, [**?**] emphasises the importance of reproduction in scientific publication: *"... a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures."*. The difficulty of replicating test beds contributes greatly to the lack of reproducibility of results. Thirdly, building such physical (and to a lesser extent virtual) environments and extensive testing may be very time consuming with manual configurations, and consequent risks of human error, resulting in a long feedback loop between the experiments. Consequently, the automation of tests is mostly non-trivial with many 'moving parts' on different machines. Our own experience is that a considerable amount of time/effort are spent on the setup of the test bed rather than the objectives of the research.

Table 1: Test bed characteristics

	Simulator	Physical	Virtual	Container
Functional testing	X	✓	✓	✓
Scalability	✓	X	?	✓
Topology flexibility	✓	X	✓	✓
Low cost	✓	X	?	✓

✓=Provided, X=Not provided, ?=Provided to some extent

Network conditions are very important for QoE aspects of real time communications. For resource/time/cost reasons, research using a VM, physical or container based approach in this area requires the use of network emulators to reproduce experimental network conditions. There are numerous off-the-shelf network emulators, such as Netem, Dummynet, NISTNet, and though these emulators are powerful and reasonably matured, recent research has shown that emulators can unknowingly add significant noise to intended network profile [**?**]. We have designed a native emulator with an inbuilt validation process that further simplifies and validates the QoE testing process.

Our test bed, which is publicly available[1], is based on a bare-bones approach to container technology and provides everything available from either physical or virtual test beds but on a single machine in a more automated, scalable, reproducible, and reusable fashion. Whilst container technology has been around for many years, its suitability for testbed research has only recently been showcased, largely through the Mininet project [**?**]. In this paper, we show how a more transparent and bare-bones containersation approach will help researchers to focus on their core research by greatly reducing the implementation overhead and handling test setup in an effective manner for them. The emulator also helps researchers to achieve reproducible packet traces, hence reproducible results.

[1]https://bitbucket.org/ycinar/rtctestbed/src

10.21437/PQS.2016-30

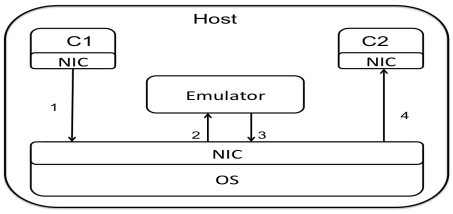

Figure 1: High level test bed architecture

The paper is organised as follows. Section 2 presents the proposed test bed methodology. Section 3 presents a case study which showcases its use and section 4 concludes the paper.

2. Methodology

The building blocks of the architecture are depicted in Fig. **??**.

2.1. Host

The complete solution resides on a single host. Containers share the kernel on the host, which is a core principle of containers [**?**]. The emulator runs on the host directly, without containers.

2.2. Containers (C1 and C2,... Cn)

Container technology is a well studied area of computer science in the recent years. It is experiencing high adoption rate due to its many advantages, such as isolation, security, scalability, performance, etc. The concept of containers is actually quite old and based on the isolation of resources, e.g. file system, network interface, etc. An important breakthrough in containers history was the Linux Containers (LXC) project by Google in 2008 [**?**].

In Fig. **??**, C1 and C2 are containers in which the MM software runs. Each of C1 and C2 possesses its own network interface, hence network isolation is ensured. This thus mimics the traditional test bed, where each MM application runs on a different machine.

One key strength of a container based approach is that hundreds of containers can easily be run on a single machine. For instance, according to a test[2] on a machine with 16GB RAM, 536 containers were run where only 37 virtual machines could be hosted. This can be very useful for researchers to reduce the cost of highly scalable experiments. For instance, a conference call experiment with 50 participants can be emulated with 50 containers representing each separate VoIP endpoint. Running 50 containers on a commodity computer is easily achievable.

Furthermore, time synchronisation is an important issue for RTC, whereby Time-of-Day synchronisation to better than millisecond can often be needed for high fidelity applications. [**?**] outlines a range of RTC applications such as Voice and Video Conferencing, Massively Multiplayer Online Gaming, Cloud Gaming, and Hybrid Broadcast Broadband TV (HbbTV/Hbb-Next) where synchronised time and timing are important to either deliver or enhance quality. There is significant effort and very detailed configuration involved in traditional multi-host test beds to synchronise time to this degree. Since containers are running on the same host, they use the same system time. Hence, they are time synchronised as best as any two processes on the same host without any additional synchronisation overhead. Although the container approach will be exposed to OS scheduling jitter, this will be significantly less than that encountered by a VM approach due to the more light-weight implementation and in any event, such jitter is typically negligible in the context of applications being tested. Note that the Mininet project [**?**] also makes use of containers under the hood, and is designed to facilitate rapid deployment of large scale complex networks with a focus on Software Defined Networking, whereas our approach is to also use containers but in a much simpler and more transparent way. Furthermore, Mininet's complex design and our need for a robust emulator and other specific requirements such as integrated objective voice quality testing, pushed us in the direction of a bare-bones transparent container-based implementation.

Container replication is almost effortless and cheap from resources perspective (time, disk space etc.), compared to physical/virtual machines. Recording, monitoring, quality evaluation tools can be part of the container to capture inputs/outputs and assess the results. A case study is presented later in the paper to showcase the concept.

2.3. Emulator

Whilst many third party emulators exist, we integrated a native emulator for ease of deployment that is also configurable with a packet trace file, i.e. data driven traffic pattern, and thus easily pluggable into the test bed. Recent research [**?**] [**?**] [**?**] has shown that emulator performance regarding delay/jitter/loss distributions can often deviate significantly from design, thus adding noise to eventual results. As such, we realise that yet another emulator may raise fidelity concerns, and thus we help address such concerns using a very simple design and ensuring that its absolute accuracy is validated by comparing the packet traces at the network interfaces. Therefore, the proposed approach includes a packet capture using tcpdump and associated code to check whether the desired network traffic was produced during the experiment. Fig. **??** presents an example output for checking the accuracy of the emulated packet trace.

Whilst popular emulators, such as Kaunet [**?**] and other alternatives [**?**], exist, none adequately met all of our needs. While they are quite powerful solutions for the problems they solve, our requirements, such as data driven traffic pattern, validation of reproduction of the desired pattern, installing/running third party software (WebRTC), and integrating objective speech quality models, required a bare-bones approach so as to keep the design relatively simple.

2.4. Quality Assessment

Ultimately, our use of containerisation is to facilitate more scalable and flexible QoS/QoE testing of RTC applications. The proposed approach offers a scalable and efficient quality assessment test bed where one can reproduce packet trace files and see the results using an objective quality assessment method, such as PESQ or POLQA. Speech quality impacting parameters that can be controlled in the test bed include codec selection as well as typical network parameters, such as delay or jitter. Our testbed fully integrates both PESQ and POLQA into the workflow along with the reference and degraded speech files to predict the MOS scores.

[2] https://insights.ubuntu.com/2015/06/11/how-manycontainers- can-you-run-on-your-machine/

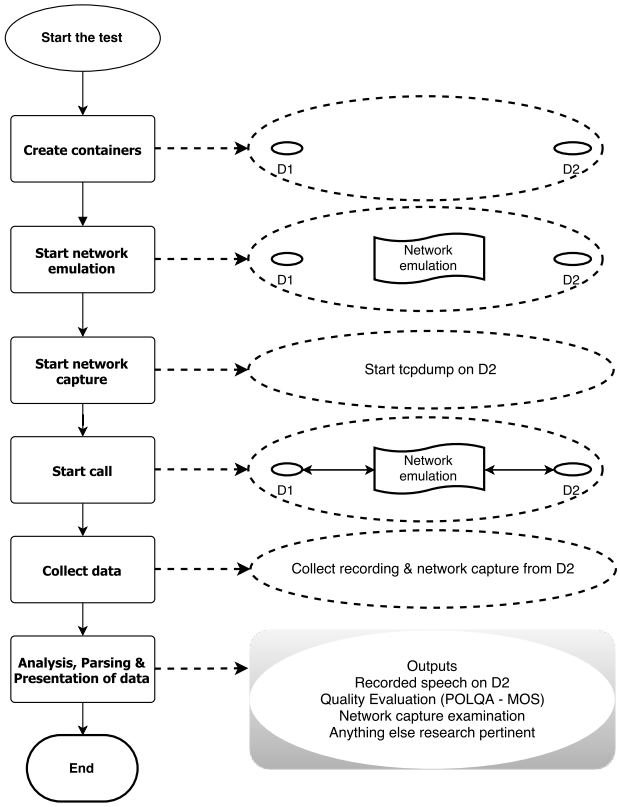

Figure 2: Test automation flow

2.5. Test Automation

A key objective of our experiments is end to end automation. While this is still possible with other test beds, it is more complicated with more uncertainty when it comes to physical (and less so virtual) test beds. Containers make this simpler since everything can be bundled in containers and run on a single machine via a relatively simple script. It is worth noting here that Mininet provides very powerful automation interfaces, however, due to the aforementioned reasons (data driven traffic pattern, validation, etc.), we have built our own test bed with a focus on speech quality evaluation.

3. A Case Study

To showcase our approach, we configured a simple test bed that assesses speech quality under real world network conditions. It is worth noting here that the proposed baseline methodology does not mandate a specific experiment or research area.

3.1. Automation

Fig. **??** shows a flowchart of the automation scripts involved in the approach. At the start of the script, the test bed is prepared by creating the containers which represent each VoIP endpoint. Network emulation to manipulate the network conditions and packet monitoring/capturing process is started at next steps. Then, the actual experiment, the VoIP call session, is started and continued for a configured duration. When the call finishes, the outputs, such as packet dump and speech, are collected, parsed, analysed, and put in a presentable format.

3.2. Experiment and Results

As part of a more comprehensive project investigating packet scaling of speech transmission using the WebRTC application, we have gathered packet traces from real world deployments. We then ran experiments reproducing the network conditions, i.e. according to the collected packet traces, using our containers based test bed. One container represents the sender/speaker side of a VoIP session and other represents the receiver/listening side. The outputs include the recording of the voice stream and tcpdump from the receiver. The reference voice stream and recording from the receiver are provided, in a form of the required audio format, to the objective speech quality prediction model, i.e. PESQ and/or POLQA, to get a MOS score for the corresponding transmission. The automation script for the experiments draws the waveforms for the reference and recorded audio files as can be seen in Fig. **??**. For this pair of the files, a MOS score of 4.134 was predicted by PESQ.

It is also important to verify if the desired network conditions were accurately reproduced. Fig. **??** shows the performance of the emulator whereby red represents the expected (configured) packet trace and blue represents the actual values according to the packet trace of the experiment. The correlation between the actual packet traces and desired packet traces is, in this case, 0.997 and reflects the accuracies we have seen through extensive testing.

The purpose of this specific experiment was to showcase the abilities of the proposed container-based approach investigating the effects of network conditions, jitter in particular, on the WebRTC speech quality. More experiment results, for many different network jitter profiles, can be found at the *experiments*

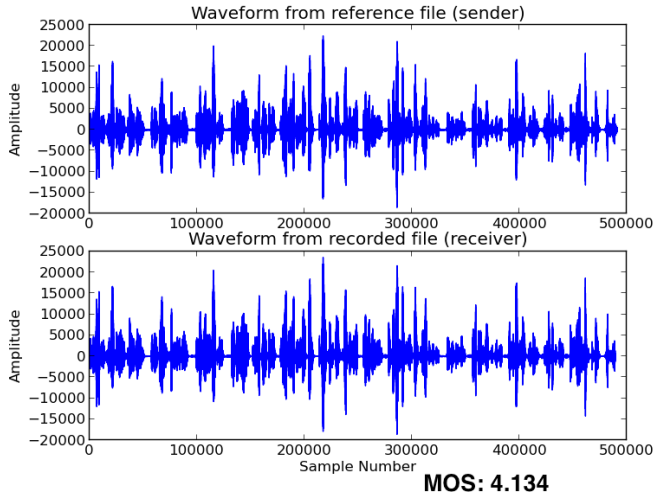

Figure 3: Waveforms

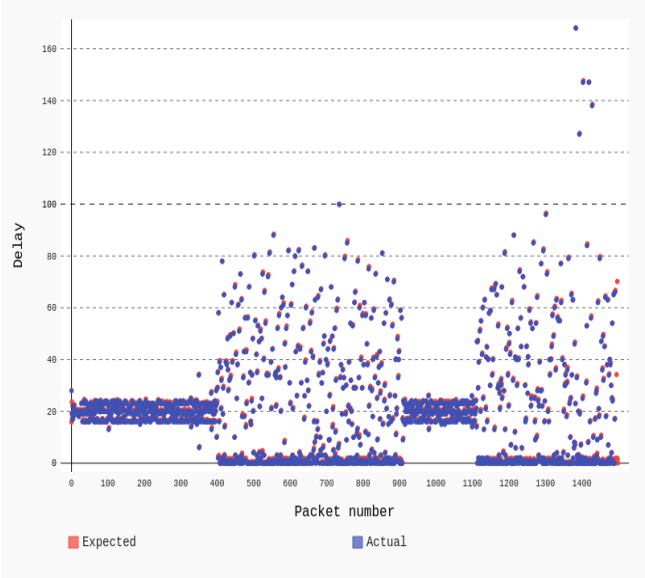

Figure 4: Emulator accuracy - time deltas

directory of the source code of the test bed[3].

4. Conclusion

In this paper, we propose a new approach for multimedia quality centric research. The implementation of the approach is open sourced and utilises container technologies as well as some other utilities.

We showcased the multidimensional advantages of containerisation of test bed for RTC quality research compared to either physical or virtual machine based approaches. Containers in RTC quality centric studies can help researchers to run their quality experiments in a more effective manner by setting up complex test beds faster and automating the experiments, whilst also improving reproducibility. This allows them to focus more on their research questions rather than unrelated tedious and cumbersome configuration issues. Containers based approach is also less resource hungry as opposed to the classical approaches, which uses physical or virtual machines. Moreover, the approach included a native/embedded network emulation technique with a verification mechanism, which allows one to have a granular and full control, i.e. packet by packet, of the network.

5. Acknowledgement

This work has been partially supported by the ICT COST Action IC1303 - Autonomous Control for a Reliable Internet of Services (ACROSS), November 14, 2013 – November 13, 2017, funded by European Union.

6. References

[1] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Reproducible network experiments using container-based emulation," in *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 2012, pp. 253–264.

[2] J. B. Buckheit and D. L. Donoho, *Wavelab and reproducible research*. Springer, 1995.

[3] T. Hofeld and M. Fiedler, "The unexpected qoe killer: When the network emulator misshapes traffic and qoe," in *Quality of Multimedia Experience (QoMEX), 2015 Seventh International Workshop on*, May 2015, pp. 1–6.

[4] B. Lantz, B. Heller, and N. McKeown, "A network in a laptop: rapid prototyping for software-defined networks," in *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*. ACM, 2010, p. 19.

[5] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio, "An updated performance comparison of virtual machines and linux containers," in *Performance Analysis of Systems and Software (ISPASS), 2015 IEEE International Symposium On*. IEEE, 2015, pp. 171–172.

[6] A. Mouat, *Using Docker*. O'Reilly Media, Inc., 2015.

[7] H. Melvin, K. Stanton, and S. Chandoke, "Time-awareness for mediasynchronisation - opportunities and challenges," in *MediaSynch Workshop, ACM TVX*, 2015.

[8] L. Nussbaum and O. Richard, "A comparative study of network link emulators," in *Proceedings of the 2009 Spring Simulation Multiconference*, ser. SpringSim '09. San Diego, CA, USA: Society for Computer Simulation International, 2009, pp. 85:1–85:8. [Online]. Available: http://dl.acm.org/citation.cfm?id=1639809.1639898

[9] H. Melvin and M. Alahmadi, "Network emulators - can they be trusted?" in *COST Action IC1304 Qualinet - Workshop*, 2014.

[10] J. Garcia, P. Hurtig, and A. Brunström, "Kaunet: A versatile and flexible emulation system," in *Swedish National Computer Networking Workshop (SNCNW08)*, 2008.

[3] https://bitbucket.org/ycinar/rtctestbed/src/0247d49f72837d97e0b5bedd992f3cbc428db30f