2023 15th International Conference on Quality of Multimedia Experience (QoMEX)

ALTRUIST: A Multi-platform Tool for Conducting QoE Subjective Tests

Henrique Souza Rossi[1], Karan Mitra[1], Christer Åhlund[1], Irina Cotanis[2], Niclas Ögren[2], Per Johansson[2]
[1]Mobile and Pervasive Computing, Department of Computer Science, Electrical and Space Engineering, Luleå University of Technology, Skellefteå, Sweden
[2]Infovista, Skellefteå, Sweden
Email: henrique.souza.rossi@ltu.se

Abstract—**Quality of Experience (QoE) subjective assessment often demands setting up expensive lab experiments that involve controlling several software programs and services. In addition, these experiments may pose significant challenges regarding management of testbed software components as they may have to be synchronized for efficient data collection, leading to human errors or loss of time. Further, maintaining error-free repeatability between subsequent subjective tests and comprehensive data collection is essential. Therefore, this paper proposes, develops and validates ALTRUIST, a multi-platform tool that assists the experimenter in conducting subjective tests by controlling external applications, facilitates data collection and automates test execution for conducting repeatable subjective tests in broad application areas.**

Index Terms—**Subjective tests, Quality of Experience, Experiments, Toolkit, Multimedia, Games, Mixed Reality**

I. Introduction

The extensive progress of multimedia applications, branching into PC/Mobile/extended reality (XR) games, cloud gaming, and audio and video streaming for entertainment and communication, among many other forms, create new opportunities and challenges in studying QoE. In particular, carrying out subjective tests in the context of the applications mentioned above usually involves setting up an extensive lab environment hosting a variety of devices that support the intended studied media. Subjective tests involve collecting a diverse set of data, ranging from questionnaires, network traffic, video traffic, physiological data, and other application-specific data logs, and most often all in parallel.

It is paramount to follow a rigorous quality data collection approach for logging, by cataloguing relevant timestamps, user/test ID, along with other variables that pertain to a particular test. This imposes challenges to the experimenter conducting the experiment, as it depends on his/her acuity to guarantee no mistakes are made. Otherwise, they may easily translate into experiment data loss or reduced data quality. Previous works have proposed tools to facilitate QoE experiments and data collection in the context of passive 2D video testing [1], [2], and 360 videos [3], [4]. A number of studies also proposed tools to show questionnaires only for web browser [5], XR locally [6]–[8] and remotely [9]. Further, some works propose frameworks to easily manage and customize an experiment locally [10]–[12], or remotely [13], [14] – all necessitating to be integrated into the testing application (e.g., a game) via code. However, those tools do not have a simplified distributed architecture, that allows executing tasks (e.g., tracking logs, calling external scripts, or starting an external application) and exerting some controlling power over external software independently (e.g., a third-party game or streaming service). These are important when studying the impact of external software and services' influence on users' QoE.

Contribution: i. This work proposes and develops ALTRUIST, a distributed QoE testing tool for conducting subjective tests efficiently. ALTRUIST minimises data collection errors across multiple devices connected via a network in a lab environment. ALTRUIST was built in Unity[1], a game engine that allows deploying applications on multiple platforms including PC, Linux, Mac, Android, and others. We leverage this feature, to create a system that can be deployed on different devices, as a single application, capable to communicate among them following a client/server network architecture; and ii. we release ALTRUIST[2] as an open-source, customizable, distributed testing tool for QoE assessment, along with a framework that can be followed to extend its applicability.

II. Motivation and System Requirements

The engineering process for building ALTRUIST emerged from the requirements of a QoE subjective study we previously conducted, in the context of mobile cloud gaming (MCG) [15]. We emulated several scenarios regarding users playing a commercial game, streamed by a commercial service to a smartphone under diverse network conditions. The following requirements were set for successfully conducting the experiment:

1) A variety of devices and platforms would be present in the lab such as two Windows PCs, three Linux PC, and an Android tablet.
2) A diversity of data logs would need to be collected for each user test which includes questionnaire, game activity, network packet capture, and streaming statistics.

[1]https://unity.com/
[2]https://github.com/hsr-research/ALTRUIST

The location where data was generated also varied across their list device.

3) Cataloging the collected data by registering the test ID, network degradation emulated, and date-time.
4) Performing a variety of tasks namely, keeping track of the network conditions list (30 in total); executing netem[3] scripts and passing them the correct network parameters; starting/stopping a game based on the match status.

In addition, items 2,3, and 4 should be done for each user test, following a systematic approach, to guarantee the same experiment conditions each time. Our team concluded that performing those tasks manually would be prone to human errors, inconsistencies, delays and data loss. To fulfil our research goals given the testing context, we developed ALTRUIST to automatically manage and support our experiments.

III. ALTRUIST System

To account for the variety of devices and tasks in the lab, the ALTRUIST was designed to be portable and distributed. It was built as a Unity game engine project and leverages Unity's capacity to deploy applications on multiple platforms. The communication between each deployed application follows a server/client architecture and is coded using the network library Mirror [4]. As such, two categories of applications were developed within the system:

- **Server:** One application was developed to be the server, and responsible to receive new client connections, synchronising their global variables, receiving and broadcasting their notifications, and executing the testing mode (automatic or manual). We name this application the **ServerManager**, and it has a UI that controls the test and displays currently connected clients.
- **Clients:** Multiple applications were developed as clients and deployed in a variety of devices to perform the necessary test requirements. We named them **Containers** since they hold a set of agents. The current list of supported agents is presented in Table I. Agents perform tasks such as: opening a process (e.g., a questionnaire app, or game), and copy/delete a file (e.g., game streaming logs, network traffic). QTnaireTask only defines the tasks (e.g., open/close) given there are many questionnaire tool-kit [6]–[8] that can show a questionnaire UI system in Unity. All the other agents provide both tasks and full code to execute them.

The system's applications can be deployed to multiple supported Unity platforms (Windows, Linux, Android etc). In the lab environment where the experiment of [15] was conducted, the following platforms were chosen to deploy each application of the system (see Fig. 1): PC1 and PC4 (Windows 10) hosted the ServerManager and Game-Streaming tasks; PC2 and PC3 (Ubuntu) hosted the Network-Emulation and Game-Server tasks respectively; an Android tablet (TB) hosted the Questionnaire tasks. A computer can host one or many clients' Container apps. However, only one ServerManger must exist in the network.

[3]https://wiki.linuxfoundation.org/networking/netem
[4]https://mirror-networking.com/

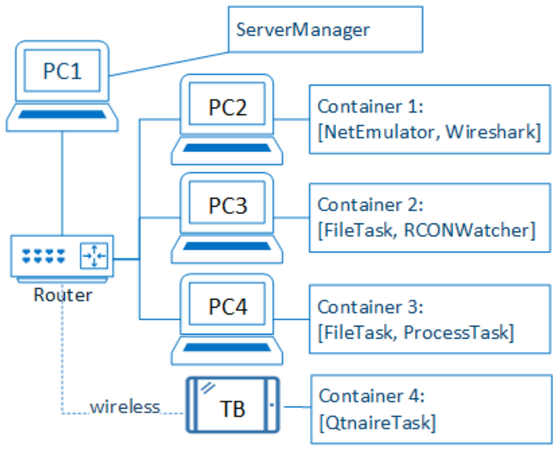


Fig. 1: Describes the use case ALTRUIST was tested. Each Container app communicates with ServerManager via TCP.

TABLE I: ALTRUIST's system agents and their tasks.

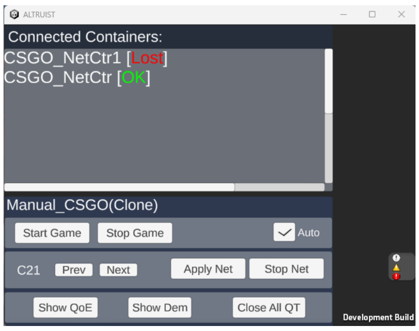
Agent Name	Task	Agent Name	Task
Wireshark	Start/Stop Wireshark network traffic capture.	NetEmulator	Applies network conditions using Netem.
Ping	Send ICMP packets and tracks RTT, PL, or Jitter.	PortWatcher	Watches or send UPD/TCP port messages.
FileTask	Copy, delete or rename files in the system.	WinResources	Tracks GPU,CPU, RAM of a proccess.
ProcessTask	Starts an application or kills a process.	RconWatcher	Reads/Sends Message to apps that supports RCON[5].
QTnaireTask	Open/Close/Save Questionnaire answers.		

Creating new Containers, or updating existing ones is performed by editing a json file which defines a single Container along with its list of agents and their settings. There are no limit to the number of agents a Container can execute. Once defined, the json file must be placed in ALTRUIST app folder for the clients that will use the Container. Starting the system as a client and loading a particular Container is done by passing arguments to ALTRUIST.exe "-t ContainerName" or as the server using "-t ServerManager". This process can be done without compiling the project from Unity editor.

Triggering the agents to perform their tasks, requires calling a single function via C# code: *SendAction-ConnectedDevices(ActionName, ContainerName, AgentName)*. This code can be called by a button from Unity canvas system. In this way, many agents can simultaneously start/stop their tasks with the press of a button. There are many examples of these use cases contained within the project and documentation.

When a client Container starts, its behaviour is to activate a local network discovery script, in an attempt to find a suitable server to connect automatically. This communication is done following the UDP protocol. Once a server is found a connection is established and TCP protocol is used for sending messages between server and clients. Those messages can be server actions to clients or notifications of events sent from

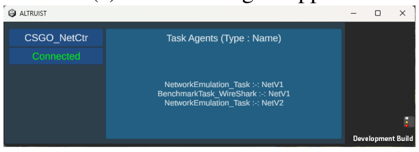
[5]https://developer.valvesoftware.com/wiki/Source_RCON_Protocol



Connected Containers:
CSGO_NetCtr1 [Lost]
CSGO_NetCtr [OK]

Manual_CSGO(Clone)

(a) ServerManager App.



CSGO_NetCtr
Connected

Task Agents (Type : Name)

NetworkEmulation_Task :-: NetV1
BenchmarkTask_WireShark :-: NetV1
NetworkEmulation_Task :-: NetV2

(b) Client Container App.

Fig. 2: System's default applications.

clients to a server. It is also possible to establish a remote connection provided the ServerManager machine IP address can be reached, and network port 7777 are unblocked. To do so ALTRUIST clients should start with the special argument "-c True".

All the system's applications have an UI that displays informative settings to be changed on-the-fly. The ServerManager UI Fig. 2a shows all the connected devices, while clients UI shows a list of the agents loaded to perform tasks Fig. 2b. For the apps that have the UI on, it is possible to show the system's console messages on screen via a free Unity plugin.

IV. Performance Impact

We assessed the performance of the ServerManger app by collecting resource consumption of the application process executing in a Windows 10 computer. The hosting computer was equipped with an Intel i7-8700 CPU, a RTX 2070 GPU, and 32 GB of RAM. The results are reported in Table II, and indicates, in its basic form, ALTRUIST is lightweight with low overall consumption, and it could fit well in even lower computer specs.

TABLE II: ServerManager app's resource consumption report after executing in a Windows PC, 30 times for 90s each.

Application	**CPU (%) Min/Max/Avg**	**GPU (%) Min/Max/Avg**	**RAM (MB) Min/Max/Avg**
Server App.	0.18//6.37//2.43	1.18//9.43//5.58	325//451//385

V. Discussion

QoE subjective tests commonly involve asking users to rate their opinion about using a system or service. It will most likely require showing a stimulus (e.g., a video game), and asking after-test questions. In its current form, ALTRUIST can manage those basic tasks presented in Table I out-of-the-box. Thanks to Unity multiple-platform support, the system can be

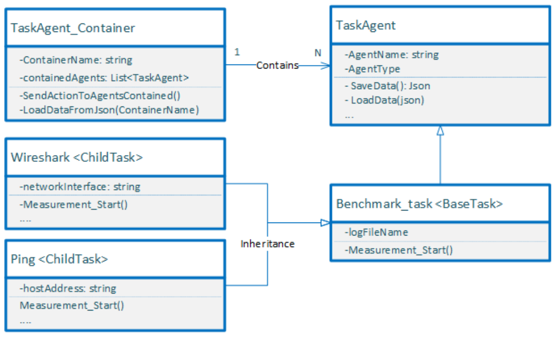


Fig. 3: System overall code's architecture. New agents should inherit from abstract class TaskAgent.

deployed to the most common platforms in the market. Since ALTRUIST follows a client/server architecture, it can be used either in local (e.g. controlled lab environment) or remote (e.g. crowd-source) experiments.

In case there is a need to extend the functionalities of the agents or create new ones, the coding process follows an object-oriented approach where each agent was written as a single C# class Unity script, inherited from a base abstract class that holds common functionalities for a group of agents. The class diagram in Fig. 3 shows the overall hierarchy using the two benchmark agents Ping and Wireshark as examples. Since they both perform a measurement task, the abstract class Benchmark-Task was created and defines common functionalities for measurement agents. The system is distributed as a Unity package, following the approach of [6], along with its source code. This allows easy integration with other Unity tools packages such as [4], [6]–[14], [16] .

Extending ALTRUIST demands creating or modifying a Unity project, and recompiling it. We acknowledge this as a limitation and to address it, our team is currently working on an upgrade that will provide support for coding-on-the-fly[6], following a game modding approach. For improving the UI, ALTRUIST could use a modular design that reads the UI basic structure from a file. Lastly, testing all the agents in different Unity platforms is also included in our checklist.

VI. Conclusion and Future Work

In the context of tools for conducting QoE experiments, we propose, develop and validate ALTRUIST, a lightweight multi-platform distributed system, that can automatically perform common tasks required to successfully conduct an experiment, and collect and catalogue the data. Its code is open source and can be used to modify existing or include new features. We foresee ALTRUIST could become very robust through the usage and inclusion of new features by the multimedia community.

In the future, we plan to add more agents to the system, support new ways to insert code and modify the UI without the need to compile the project.

[6]https://github.com/dotnet/roslyn

References

[1] S. Göring, R. R. Ramachandra Rao, S. Fremerey, and A. Raake, "AVrate Voyager: an open source online testing platform," in *2021 IEEE 23rd International Workshop on Multimedia Signal Processing (MMSP)*, Oct. 2021, pp. 1–6, iSSN: 2473-3628.

[2] C. Keimel, J. Habigt, C. Horch, and K. Diepold, "QualityCrowd — A framework for crowd-based quality evaluation," in *2012 Picture Coding Symposium*, May 2012, pp. 245–248.

[3] P. Pérez and J. Escobar, "MIRO360: A Tool for Subjective Assessment of 360 Degree Video for ITU-T P.360-VR," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, Jun. 2019, pp. 1–3, iSSN: 2472-7814.

[4] C. Cortés, P. Pérez, and N. García, "Unity3D-based app for 360VR subjective quality assessment with customizable questionnaires," in *2019 IEEE 9th International Conference on Consumer Electronics (ICCE-Berlin)*, Sep. 2019, pp. 281–282, iSSN: 2166-6822.

[5] D. Guse, H. R. Orefice, G. Reimers, and O. Hohlfeld, "TheFragebogen: A Web Browser-based Questionnaire Framework for Scientific Research," in *2019 Eleventh International Conference on Quality of Multimedia Experience (QoMEX)*, Jun. 2019, pp. 1–3, iSSN: 2472-7814.

[6] G. Regal, R. Schatz, J. Schrammel, and S. Suette, "VRate: A Unity3D Asset for integrating Subjective Assessment Questionnaires in Virtual Environments," in *2018 Tenth International Conference on Quality of Multimedia Experience (QoMEX)*, May 2018, pp. 1–3, iSSN: 2472-7814.

[7] M. Feick, N. Kleer, A. Tang, and A. Krüger, "The Virtual Reality Questionnaire Toolkit," in *Adjunct Publication of the 33rd Annual ACM Symposium on User Interface Software and Technology*. Virtual Event USA: ACM, Oct. 2020, pp. 68–69.

[8] R. Tamaki and T. Nakajima, "Shoot Down Drones with Your Answer, an Integration of a Questionnaire into a VR Experience," in *Symposium on Spatial User Interaction*. Virtual Event USA: ACM, Nov. 2021, pp. 1–2.

[9] R. Bovo, D. Giunchi, A. Steed, and T. Heinis, "MR-RIEW: An MR Toolkit for Designing Remote Immersive Experiment Workflows," in *2022 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. Christchurch, New Zealand: IEEE, Mar. 2022, pp. 766–767.

[10] M. R. Watson, B. Voloh, C. Thomas, A. Hasan, and T. Womelsdorf, "USE: An integrative suite for temporally-precise psychophysical experiments in virtual environments for human, nonhuman, and artificially intelligent agents," *Journal of Neuroscience Methods*, vol. 326, p. 108374, Oct. 2019.

[11] A. O. Bebko and N. F. Troje, "bmlTUX: Design and Control of Experiments in Virtual Reality and Beyond," *i-Perception*, vol. 11, no. 4, p. 2041669520938400, Jul. 2020, publisher: SAGE Publications.

[12] J. Brookes, M. Warburton, M. Alghadier, M. Mon-Williams, and F. Mushtaq, "Studying human behavior with virtual reality: The Unity Experiment Framework," *Behavior Research Methods*, vol. 52, no. 2, pp. 455–463, Apr. 2020.

[13] A. Steed, L. Izzouzi, K. Brandstätter, S. Friston, B. Congdon, O. Olkkonen, D. Giunchi, N. Numan, and D. Swapp, "Ubiq-exp: A toolkit to build and run remote and distributed mixed reality experiments," *Frontiers in Virtual Reality*, vol. 3, p. 912078, Oct. 2022.

[14] J. Lee, R. Natarrajan, S. S. Rodriguez, P. Panda, and E. Ofek, "RemoteLab: A VR Remote Study Toolkit," in *The 35th Annual ACM Symposium on User Interface Software and Technology*. Bend OR USA: ACM, Oct. 2022, pp. 1–9.

[15] H. S. Rossi, N. Ögren, K. Mitra, I. Cotanis, C. Åhlund, and P. Johansson, "Subjective Quality of Experience Assessment in Mobile Cloud Games," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, Dec. 2022, pp. 1918–1923.

[16] S. J. Friston, B. J. Congdon, D. Swapp, L. Izzouzi, K. Brandstätter, D. Archer, O. Olkkonen, F. J. Thiel, and A. Steed, "Ubiq: A System to Build Flexible Social Virtual Reality Experiences," in *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*. Osaka Japan: ACM, Dec. 2021, pp. 1–11.