December 2019

Universal and automatic end-to-end testing of smart TVs

Ping Zhou

Amit Agarwal

Achal Pandey

Eric Yin

Universal and automatic end-to-end testing of smart TVs

ABSTRACT

End-to-end testing of smart TVs has proven to be challenging due to various reasons, such as: the proprietary nature of hardware and software from different TV manufacturers; the difficulty of identifying performance bottlenecks due to visual rendering being performed by cloud-based renderers; etc. As a result, end-to-end testing of a smart TV still relies heavily on time-consuming and costly manual testing. This disclosure describes a framework for end-to-end testing of smart TVs that is automatic, e.g., uses minimal human intervention, and universal (agnostic to TV manufacturer).

KEYWORDS

- TV testing
- Smart TV
- Universal TV
- Television
- Set-top box
- Voice assistant
- Smart assistant
- Voice query
- End-to-end (E2E) testing
- Robotic input emulator
- Voice remote controller
- Latency profile

BACKGROUND

A smart TV that has a voice interface (e.g., that includes voice assistant or other voice-activated software) provides a rich, intuitive way for users to access information, services, and to access other smart devices. However, end-to-end (E2E) testing of smart TVs has proven to be challenging due to various reasons such as:

- The voice-based experience on TV is highly interactive, with diverse audio/video responses from multiple sources. As a result, end-to-end testing of a smart TV still relies heavily on time-consuming and costly manual testing.
- Proprietary hardware and software from different TV manufacturers are effectively black boxes to a tester. Without internal knowledge provided by the TV manufacturer, programmatic control of test flow is difficult.
- On lower-end smart TV models that have limited on-board resources, visual responses are rendered by a cloud-based renderer, which is also a black box to a tester. Thus, if a user experiences a sluggish user interface, it is difficult to break down the end-to-end latency and identify the bottleneck. For example, it is not clear if it took too long for a query to reach the cloud renderer, or if the cloud renderer took too long to deliver the resultant user interface screen or other response to the TV.

Moreover, a viable test framework must also scale such that it works with devices from different manufacturers without relying on specific hardware, software, or external support.

DESCRIPTION

This disclosure describes a framework for end-to-end testing of smart TVs that is automatic, e.g., uses minimal human intervention, and universal (agnostic to TV manufacturer).

Test Architecture

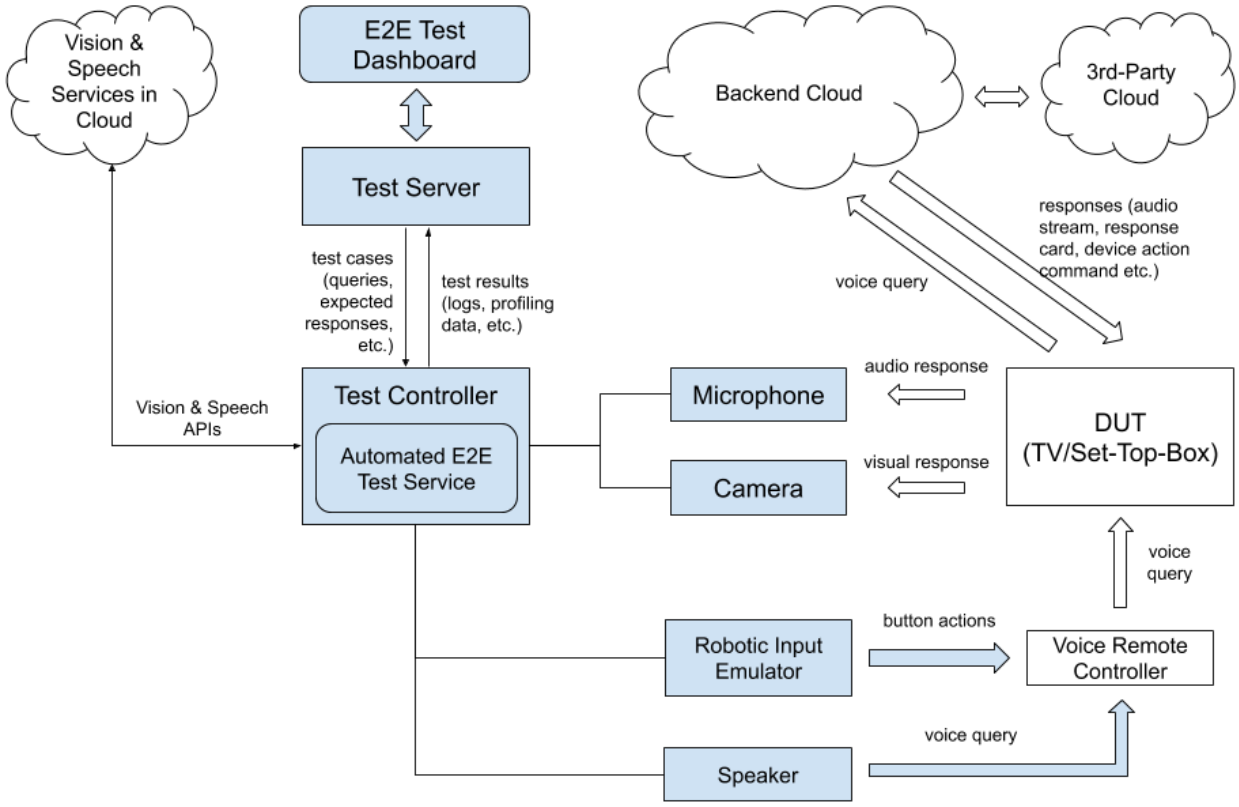

Fig. 1: Test architecture

Fig. 1 illustrates a test architecture to test televisions (or other devices), per techniques of this disclosure. The device under test (DUT) is the TV or set-top-box, which communicates with a backend cloud (e.g., provided by the TV manufacturer), which in turn may communicate with a third-party cloud (e.g., other parties that provide services). The DUT is controllable by a voice remote controller through which voice commands may be issued. Test architecture components are depicted in blue, and are as follows:

- A *control PC* is a computer equipped with peripherals and software for automated testing.
- An *automated E2E test service* is a software service that runs on the control PC and controls the automated end-to-end test flow.

- A *test server* is a local or remote server that stores test cases and results.
- An *E2E test dashboard* is a web interface that enables a tester to manage test cases and visualize test results.
- A *microphone* captures the audio response from the DUT.
- A *camera* captures the visual response from the DUT.
- A *speaker* issues voice queries to the voice remote controller paired with the DUT.
- A *robotic input emulator* (RIE) is a device that programmatically presses and releases buttons on the voice remote controller.

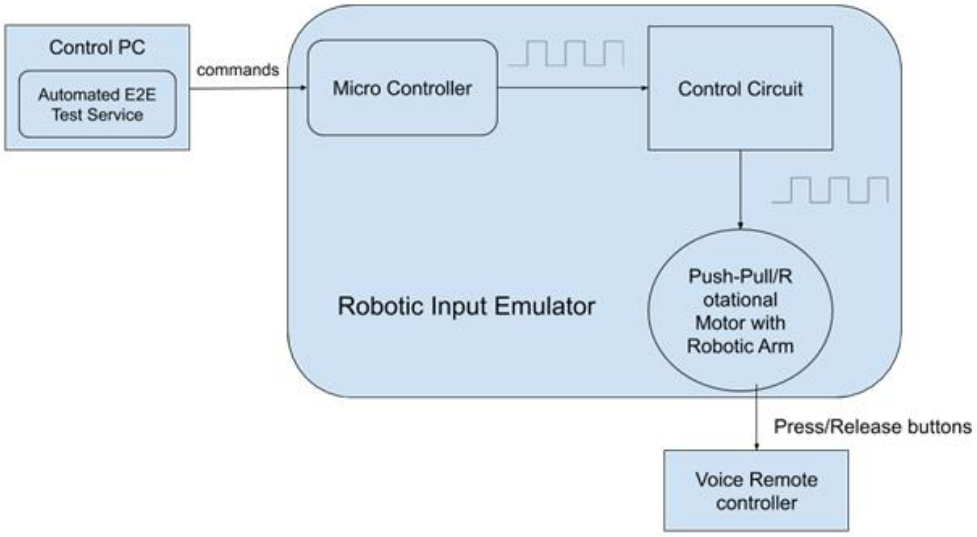

Fig. 2: A robotic input emulator

As illustrated in Fig. 2, a RIE includes a microcontroller that accepts commands from the control PC and a motorized solenoid that simulates button press/release actions. The RIE and the voice remote controller are aligned prior to testing such that the target buttons on the voice remote controller get pressed and released. A major goal is to test the voice-based E2E experience; for this purpose, a single solenoid is used for pressing and releasing the mic button of the voice remote controller. For more sophisticated interaction with the voice remote controller, multiple solenoids can be used as necessary.

Generic interaction testing

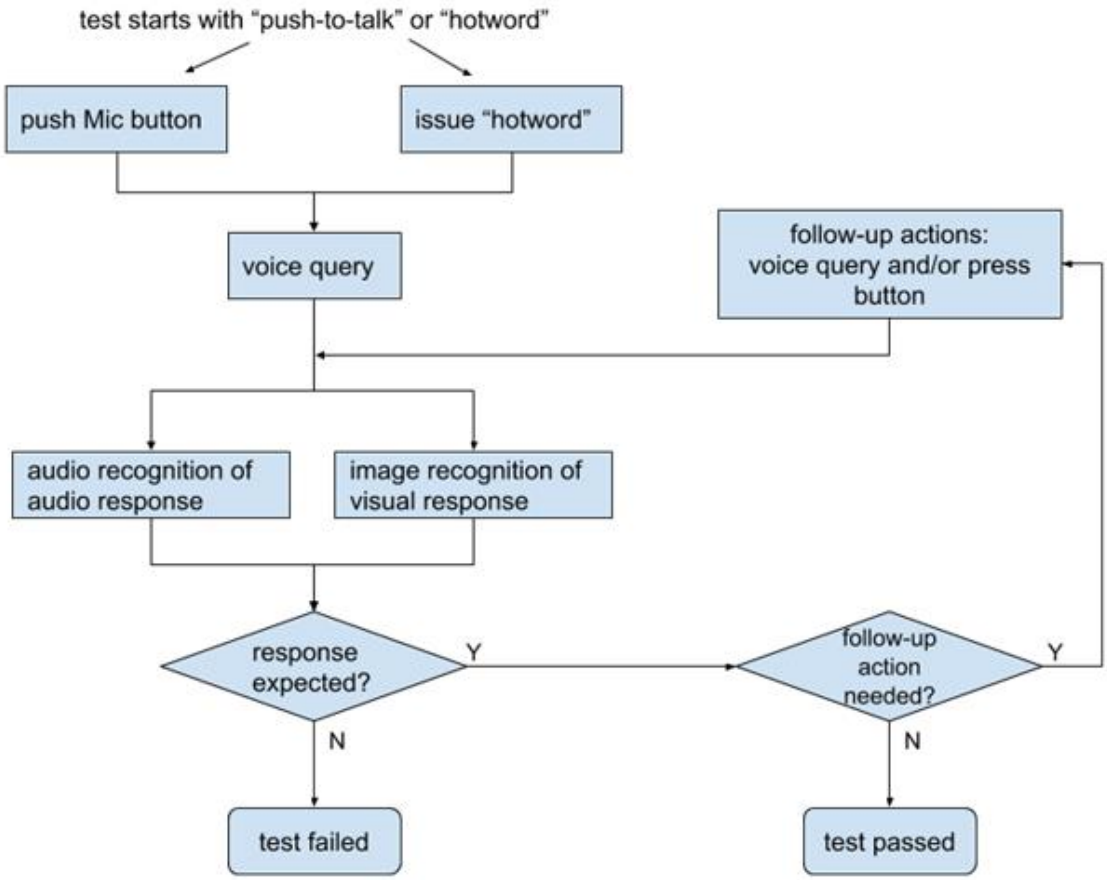

Fig. 3: Generic interaction testing

Fig. 3 illustrates the flow of end-to-end testing of smart TVs, per techniques of this disclosure. Using "what's the weather?" as an example voice query, the test flow is as follows:

- The RIE is activated to press the mic button on the voice remote controller to initiate the voice query.
- The camera and microphone are enabled to capture the responses of the DUT (smart TV).
- The test query "what's the weather" is played on the speaker.
- The mic button on the voice remote controller is released by the RIE to mark the end of the test query.

- Audio recognition and image recognition are performed to extract information from the responses of the DUT. The extracted information is compared with the expected response from a test server, and test pass/fail status is logged. In case of a time-out, the test is regarded as failed, with a time-out flag appended to the test log.
- If follow-up action is specified in the test case, e.g., a follow-up voice query such as "how about tomorrow," a new conversational turn is started where the mic button is pressed, the next query is uttered, the mic button is released, etc.
- When conversation is finished, e.g., there are no more follow-up actions, the test server is updated with test results.

Profiling end-to-end response latency

The response latency is an important metric of end-to-end user experience. The visual response returned by a backend in response to a query, usually as an embeddable HTML document, can be rendered locally by the device, or rendered by a cloud-based renderer and delivered to the device in the form of images.

E2E response latency can be caused by different factors, for example:

- The latency of delivering the voice query to the backend cloud, e.g., slow uplink.
- The latency in the backend cloud and/or third-party cloud, e.g., latencies in generating a response.
- The latency of rendering, which can be traced to one or more of the following sources:
    - In case of local rendering: On-device renderer taking too long to render the response, for example due to high CPU/memory load.
    - In case of cloud rendering: Cloud renderer taking too long to deliver the rendered pictures, caused, for example, by insufficient server capacity, higher than

expected usage, server down, network latency between cloud renderer and devices, etc.

The techniques of this disclosure provide a breakdown of E2E response latency while considering both the DUT and the cloud renderer as black boxes, e.g., without probing into either the DUT or the cloud renderer. For the purposes of providing a breakdown of the E2E latency, the following components are defined:

- A voice query for testing E2E latency that is distinct from other queries, e.g. “test end-to-end latency.”
- A distinct visual element in response to the test query. The visual element can include a barcode or QR code generated at runtime, such that it can be associated with the DUT and the current query/conversation.
- Timestamp logging in the control PC and the backend cloud, with timestamps defined as follows:
    - T0: timestamp when the control PC issues the voice query.
    - T1: timestamp when the backend cloud receives the query.
    - T2: timestamp when the backend cloud generates the visual response.
    - T3: timestamp when the visual response is sent by the backend cloud to the DUT or the cloud renderer.
    - T4: timestamp when the control PC recognizes the visual response, e.g., the barcode or QR code.

The E2E response latency can be broken down into the following parts:

- T1−T0: time to deliver the query to the backend.
- T2−T1: time to generate a response.

- T3−T2: time to deliver a response to the DUT or the cloud renderer.
- T4−T3: time to render and deliver the rendered response.

The above time metrics are collected at periodic intervals and visualized in the E2E test dashboard. When a user experiences long E2E latency, these time metrics are helpful in identifying bottlenecks along the datapath.

The following diagrams illustrate signal flows between the test equipment and the DUT that enable profiling of end-to-end latency under different scenarios.

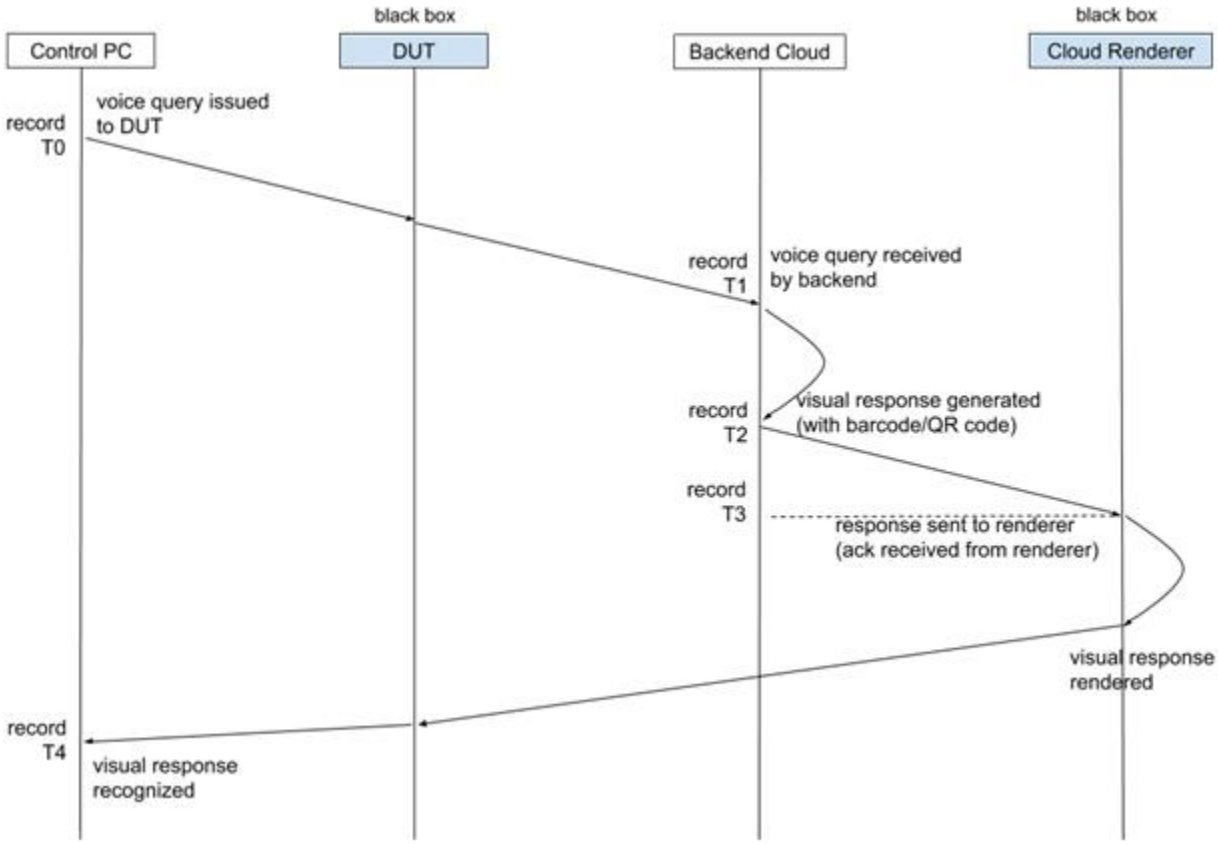

Fig. 4: Cloud rendering, with response sent directly from backend cloud to cloud renderer

Since timestamps are logged per techniques of this disclosure, it is possible to determine using the flow of Fig. 4 that the bulk of the latency is traceable to the time to deliver the query to the backend cloud (T1−T0) and the time to render and deliver the rendered response (T4−T3).

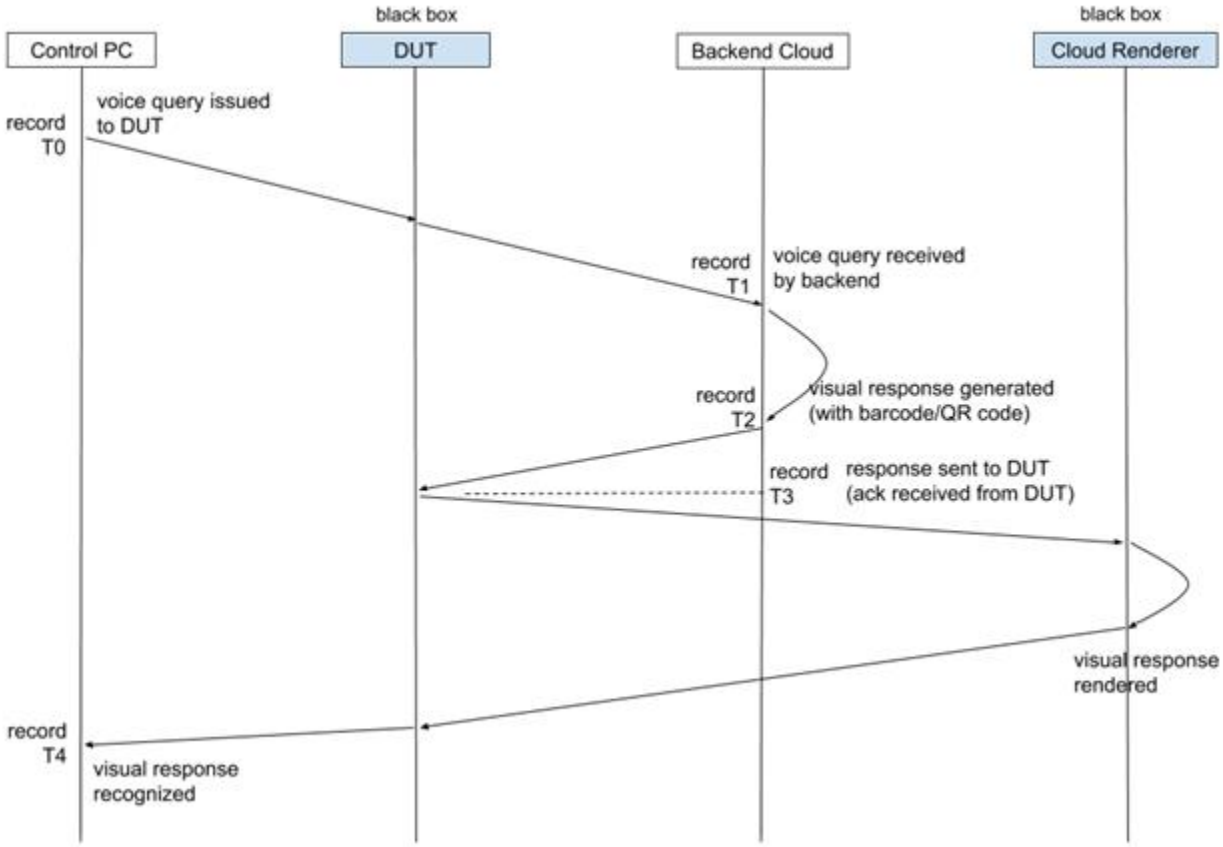

Fig. 5: Cloud rendering, with response sent from backend cloud to DUT and rendered by cloud renderer

Since timestamps are logged per techniques of this disclosure, it is possible to determine from the flow of Fig. 5 that the bulk of the latency is traceable to the time to deliver the query to the backend cloud (T1−T0) and the time to render and deliver the rendered response (T4−T3).

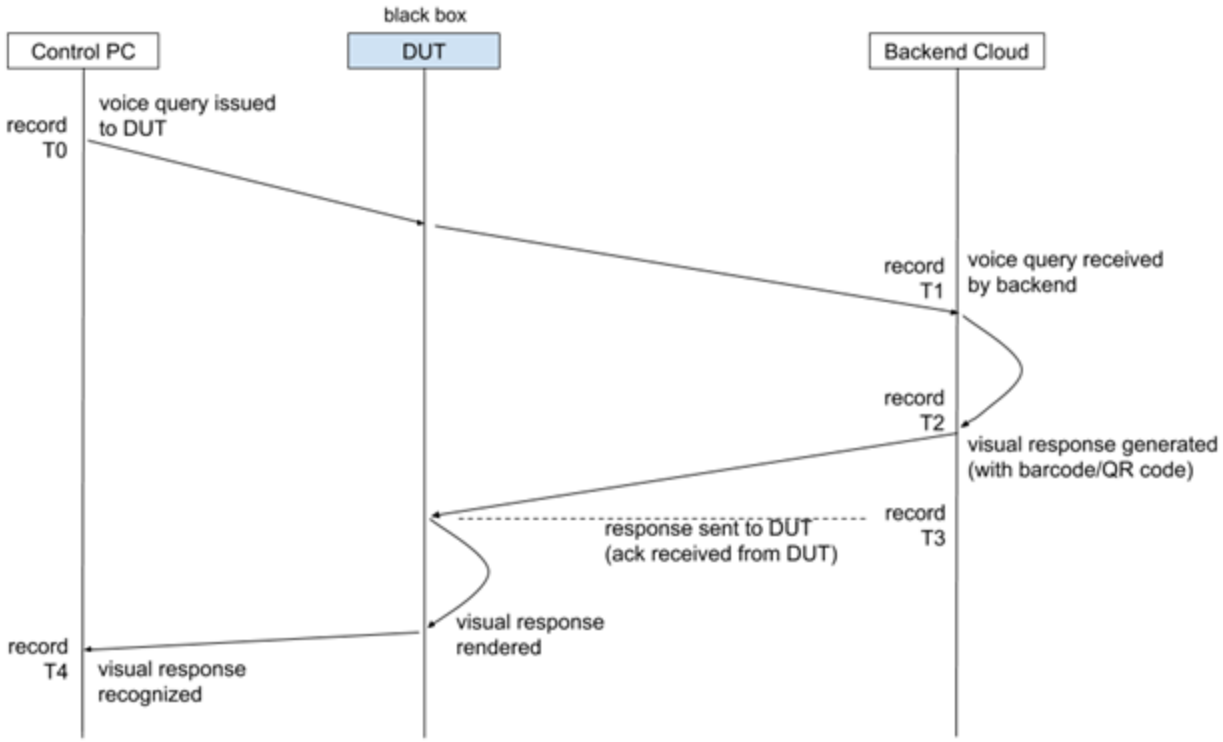

Fig. 6: Local rendering

Since timestamps are logged per techniques of this disclosure, it is possible to determine from the flow of Fig. 6 that the bulk of the latency is traceable to the time to deliver the query to the backend cloud (T1−T0), the time taken by the backend cloud to deliver a response to the DUT (T3-T2), and the time taken by the DUT to render and deliver the rendered response (T4−T3).

The techniques of this disclosure have many advantages such as:

- The end-to-end testing of smart TV, e.g., from remote controller action to voice query to audio and visual responses, is fully automated. The resulting tests simulate very closely the real end-user experience, resulting in efficiency and accuracy gains over current test methodologies, which automate only parts of the test flow. Manual operation, e.g., to

operate the TV with the remote controller, issue voice queries, and observe audio/video responses, is obviated.

- The techniques are universal, e.g., do not depend on specific hardware models (TV or remote) and flexible. The techniques do not require the adding of probes, the modifying of code in black-box components, e.g., code on the DUT, on the third-party cloud, or any other component that the tester cannot access. Therefore, the techniques can easily be utilized to test DUT from to different TV manufacturers.
- The techniques provide a non-intrusive way of profiling the breakdown of E2E latency, enabling testers or system integrators to identify bottlenecks and improve user experience. The non-intrusive nature of the test framework further enables extension to different TV manufacturers.

CONCLUSION

This disclosure describes a framework for end-to-end testing of smart TVs that is automatic, e.g., uses minimal human intervention, and universal (agnostic to TV manufacturer).