Scaling on the Edge:
A Benchmarking Suite For Human-in-the-Loop Applications

M. Olguín, J. Wang, M. Satyanarayanan, J. Gross

Abstract

Benchmarking human-in-the-loop application is complex given their nature, which heavily depends on the actions taken by the *human* user. This limits reproducibility as well as feasibility of performance evaluations. We propose a methodology and present a benchmarking suite we call Edge-Droid that can address these challenges. Our core idea rests on recording traces of these applications which are played out in a controlled fashion based on an underlying model of human behavior. The traces are then exposed to the original backend compute process of the respective human-in-the-loop application, generating realistic feedback. This allows for an automated system which greatly simplifies benchmarking large scale scenarios.

Basic Idea

- Benchmarking human-in-the-loop applications is **hard** due to *human* users:
  - They are unpredictable.
  - They make scaling difficult (you need more of them!).
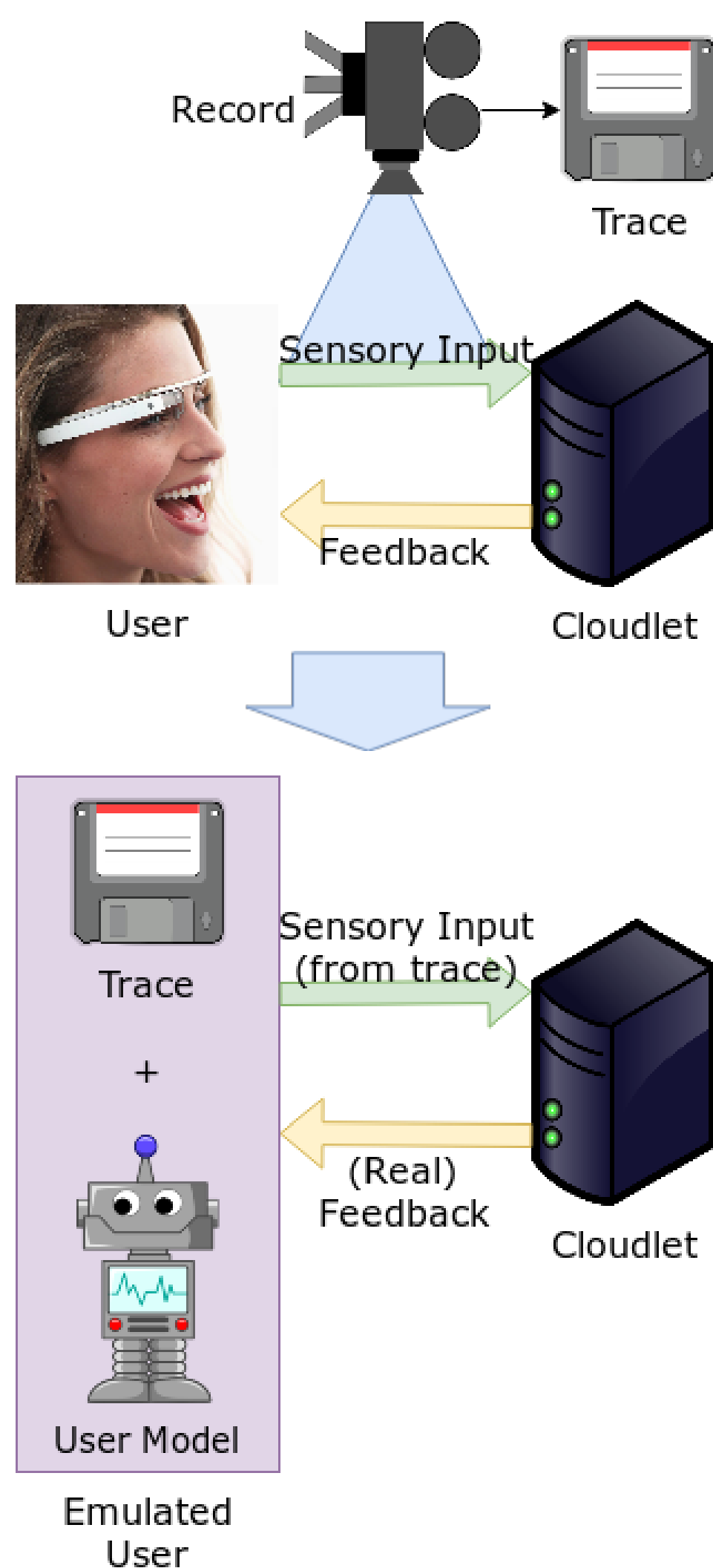- What if we could cut out the user?

Figure 1: Basic idea is to replace the user by a pre-recorded sensory input trace played through a simple user model.
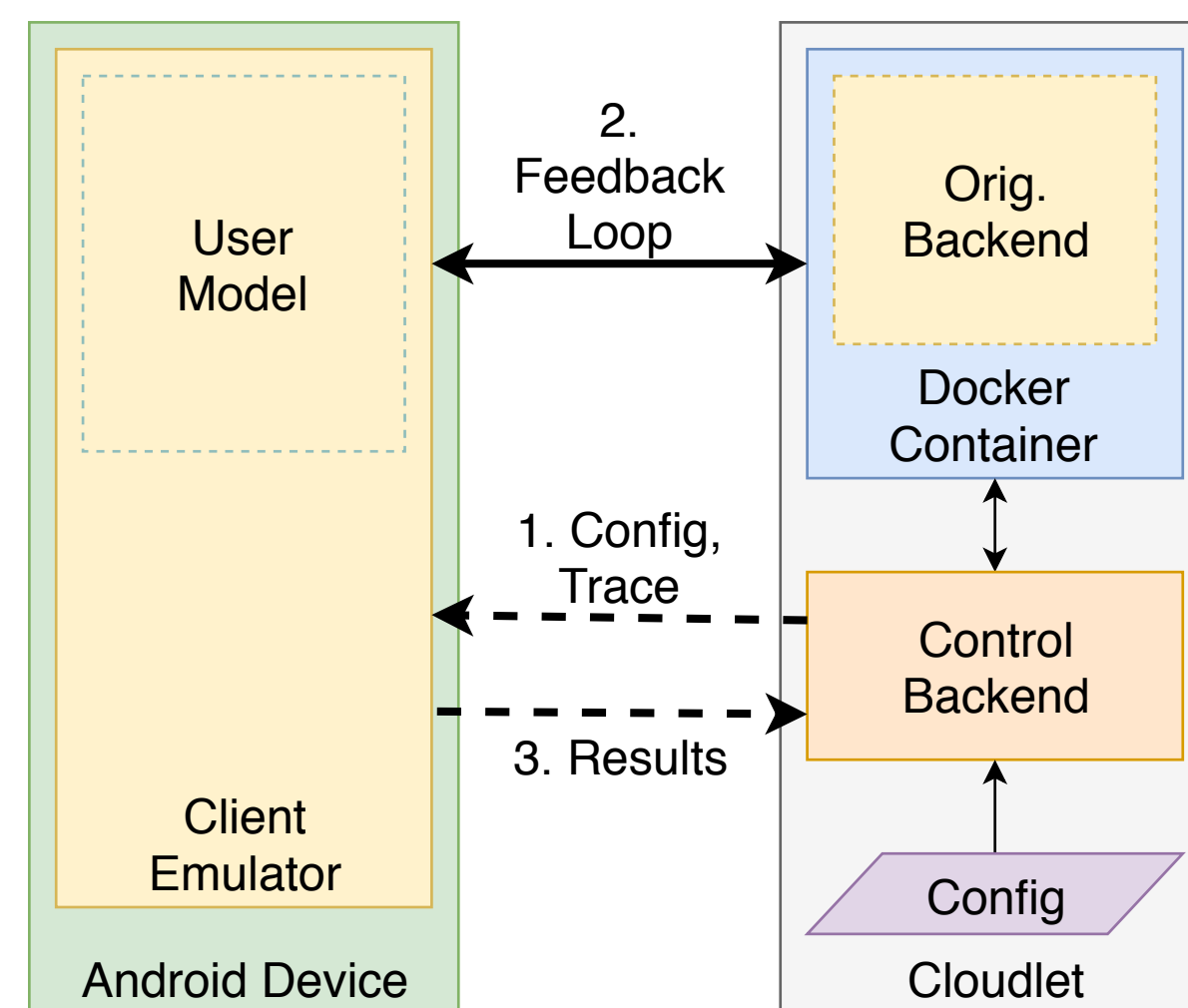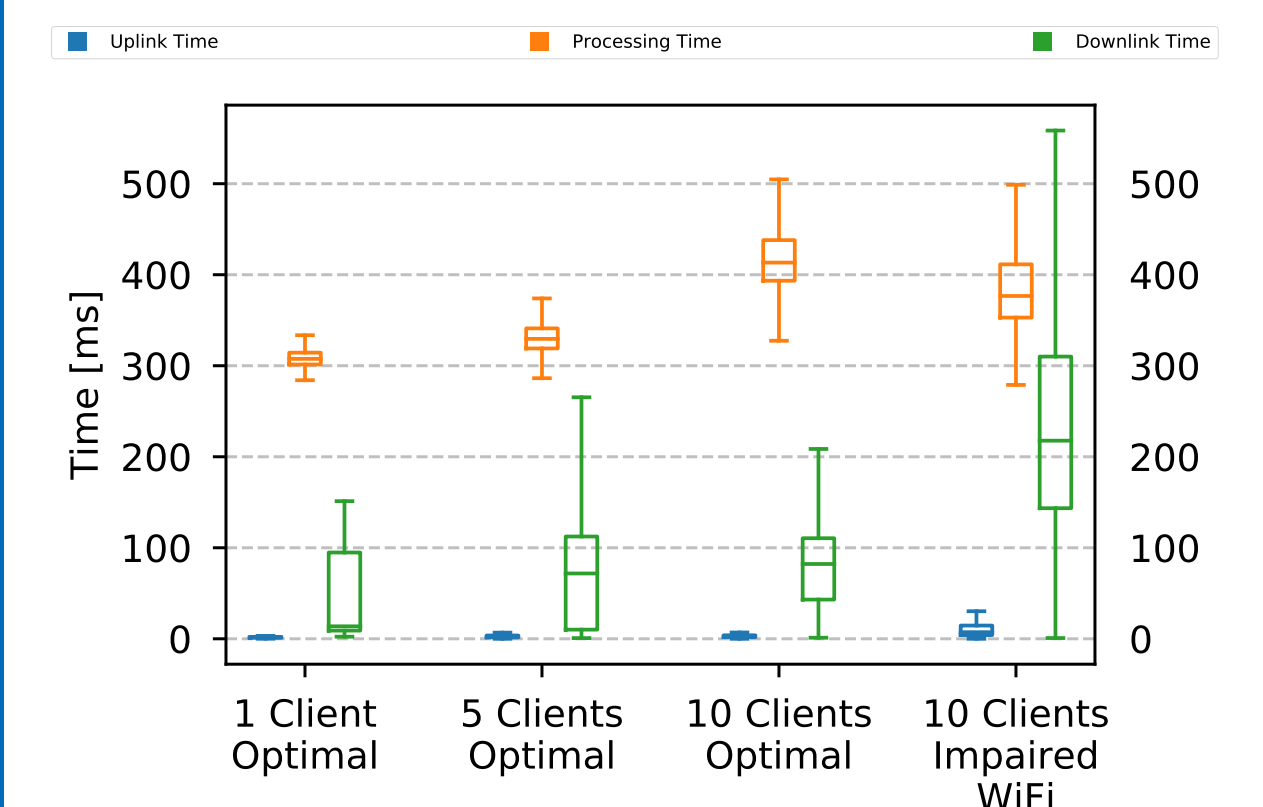
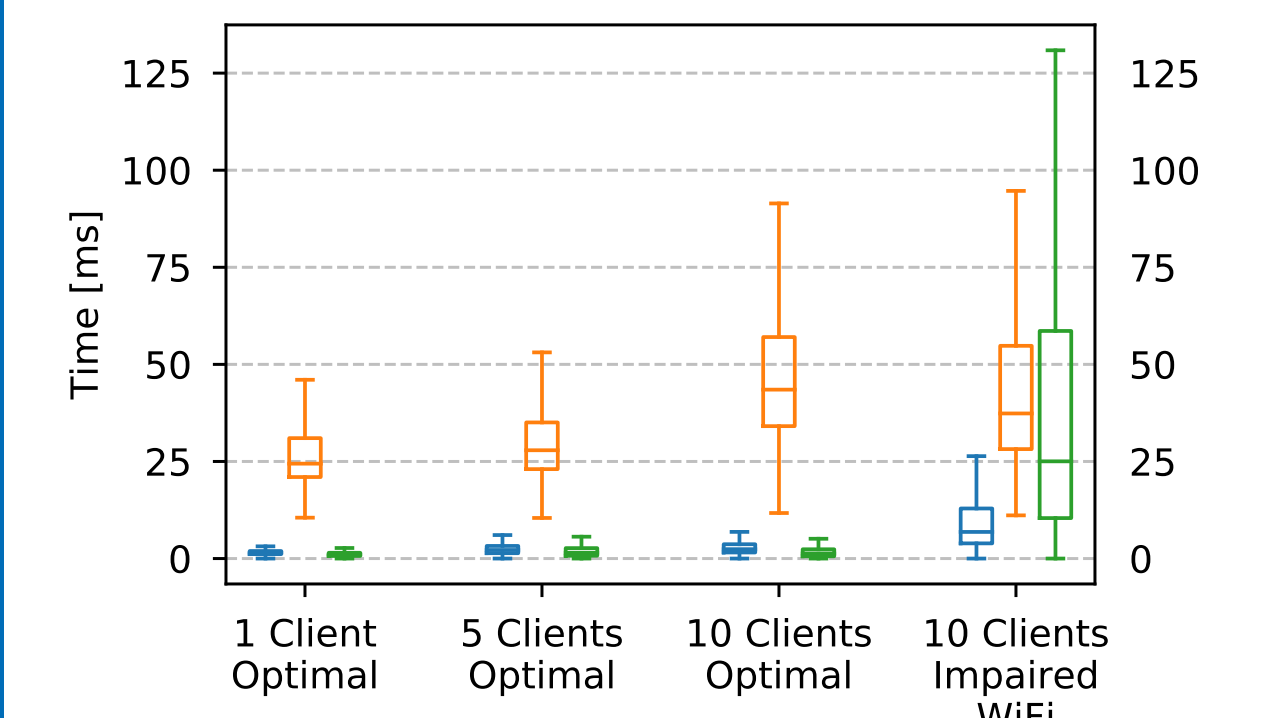## Design & Implementation

Figure 2: EdgeDroid Architecture

The architecture of our benchmarking suite consists of two main components:

- The *control backend* controls the experiments and collects measurements from the application and the cloudlet itself.
- The *client emulators* play out a prerecorded sensory input trace over the network in a controlled fashion, while collecting client-side metrics.

Figure 3: Simple user model used for the initial iteration of the suite.

Brief description of user model.

Some Example Results

a: Inputs that triggered feedback.

b: Inputs that did not trigger feedback.

Figure 4: Comparison of the latency distributions across system components for a series of scenarios, differentiated by feedback-/lack of feedback.

These results could be useful for, for instance, system designers wishing to identify bottlenecks across the system hardware stack, or for application developers to determine points of optimization in the application code.
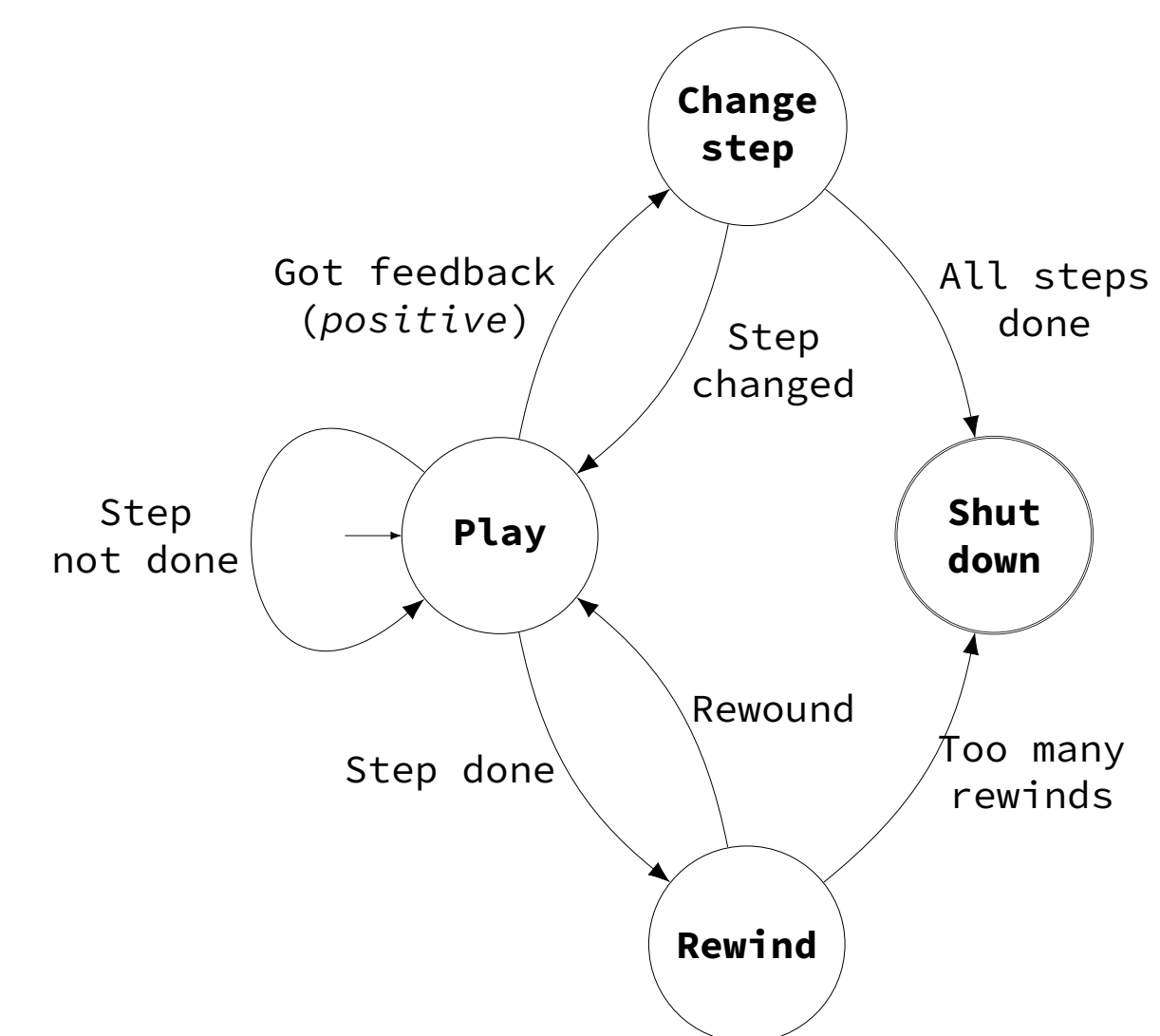
Some references here for good measure.