Poster: A Context Simulation Harness for Realistic Mobile App Testing

Manoj R. Rege, Vlado Handziski, and Adam Wolisz
Telecommunication Networks Group, Technische Universität Berlin, Germany
{rege, handzisk, wolisz}@tkn.tu-berlin.de

1. CONTEXT SIMULATION
Accurate performance testing of mobile apps require comprehensive simulation of real world context in a mobile emulator viz. location, sensor values, network conditions etc. Existing mobile emulators support such simulation, however they lack the capability for automated generation of realistic, correlated, and dependent context traces. As a result, the burden of their generation and injection is left to the developers who have to create their own traces. This is not straightforward: there are heterogenous remote databases of traces, mathematical models and trace files can be used as well, but the trace values should be correlated, and traces have to be converted to a common format. We are developing contextmonkey - a framework that addresses these concerns by leveraging context traces from databases such as Foursquare [1], Opensignal [2], Google Street View [3]. It acts as a harness to mobile emulators, and aims at improving the efficiency of mobile app performance tests.

2. DESIGN AND IMPLEMENTATION
The contextmonkey design is data-source and emulator independent, modular, and is easily extensible to new context simulation scenarios. Its architecture (shown in figure 1a) is organized into three layers:

1) Trace Layer orchestrates extraction of correlated traces from various data-sources. In case of external databases, this includes: handling authorization and authentication for using database APIs, configuring correct parameters for API calls to obtain correlated context trace values, and modifying received trace value units and attributes if necessary. For example, an image trace might need modification of resolution, and file format attributes. It also handles failures due to trace unavailability by interpolating or extrapolating their values, and from exceeded API usage limits of the databases by signaling them to the higher layer. Finally, it converts traces from database specific formats to an internal representation format and forwards it to the layer above.

2) Context Layer specifies the context modality definitions: physical sensor sampling rates, their correlations, virtual sensor functions over different physical sensor values etc. It generates trace requests using these definitions and forwards them to the Trace layer. It computes virtual sensor functions based on the received trace values. Accordingly, it temporarily stores each received trace value copy in a queue to account for delay variations in fetching them. Finally, it forwards all context trace values to the higher layer.

3) Feed Layer provides the glue to bind the framework to the specific emulator. It converts the received context trace values to the emulator specific format, and makes API calls to the emulator to feed these values.

The contextmonkey prototype integration with Android emulator using the emulator Telnet service is shown in figure 1b. The emulator implements a QEMU Daemon service, and clients that send the context trace values to the Android Goldfish Kernel which makes them available to the app. We have integrated the Google Street View database with the prototype to fetch location dependent images, and feed them to the emulator camera. We are currently in process of evaluating the prototype performance.

3. REFERENCES

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s). Copyright is held by the author/owner(s).
MobiSys ’15, May 19–22, 2015, Florence, Italy.
ACM 978-1-4503-3494-5/15/05.
http://dx.doi.org/10.1145/2742647.2745913.