Scalable Mockup Experiments on Smartphones using SmartLab

Georgios Larkou*, Marios Mintzis‡, Panayiotis G. Andreou*,
Andreas Konstantinidis* and Demetrios Zeinalipour-Yazti*

*Department of Computer Science, University of Cyprus, 1678 Nicosia, Cyprus
‡Department of Computer Science, University College London, WC1E 6BT London, UK
glarkou@cs.ucy.ac.cy; marios.mintzis.14@ucl.ac.uk; {panic, akonstan, dzeina}@cs.ucy.ac.cy

***Abstract*—In this paper we present a comprehensive architecture to carry out experimental repeatability studies on clusters of smartphones. Our architecture is founded on *SmartLab*[1], our in-house architecture for managing real and virtual smartphones via an intuitive Web user interface. Our presented architecture consists of several exciting components for re-programming and instrumenting smartphones to perform application testing and data gathering in a facile manner, as well as executing mockup experiments by "feeding" the devices with GPS/sensor readings. We will particularly demonstrate the various components of our architecture that encompasses smartphone sensor data collected by mobile users and organized in our distributed NoSQL document store. The given datasets can then be replayed on our testbed comprising of real and virtual smartphones accessible to developers through our Web 2.0 user interface. We present the applicability of our architecture through various mockup experiments over different application scenarios.**

Index Terms — mockups, testbeds, mobiles, software testing.

I. Introduction

The continuous improvements of smartphone devices and embedded sensor systems during the past decade have enabled researchers to explore complex interdisciplinary areas (e.g., behavioral sciences, social sciences) from the big data perspective. This facilitates understanding of the physical world at an extremely high fidelity and interpretation of real-life problems by analyzing individual behavior through mobility, communication and interaction patterns. The latter patterns can be obtained from modern smartphone devices that continuously provide more efficient means for big data generation through their enhanced computing and multi-sensing capabilities.

Smartphone users are constantly moving and sensing thus generating large amounts of data contributing to the evolution of new services and applications [1], also known as crowdsourcing, which is gradually becoming the prevalent mean of data gathering. Re-programming smartphones and instrumenting them for application testing and data gathering at scale is currently a tedious, time-consuming process that poses significant logistical challenges.

To this end, we have implemented *SmartLab* [2], a comprehensive architecture for managing a cluster of both *Android Real Devices (ARDs)* and *Android Virtual Devices (AVDs)*, which are managed via an intuitive web-based interface. Our current architecture is ideal for repetition of scenarios that require *fine-grained* and *low-level* control over real smartphones [3], [4], e.g., OS, Networking, DB and storage, security, peer-to-peer protocols, but also for scenarios that require the engagement of physical sensors and geo-location scenarios.

[1]Available at `http://smartlab.cs.ucy.ac.cy/`

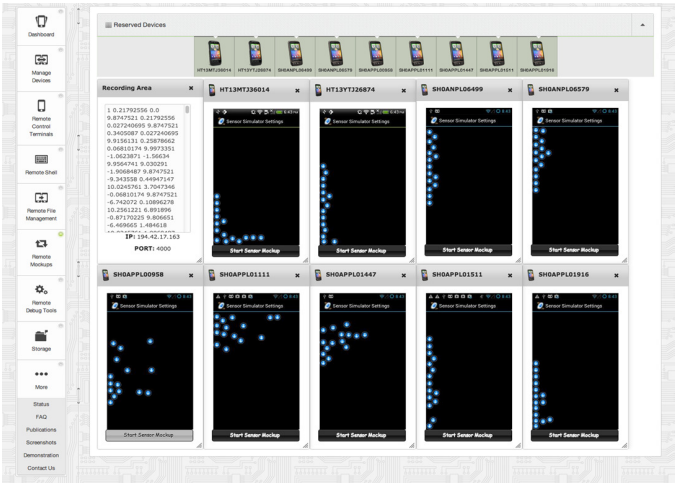

Fig. 1. **An example mockup experiment in SmartLab, where a user feeds real smartphones with sensor readings from a big-data store and overviews the results through a Web 2.0 User Interface.**

SmartLab has been inspired by *PlanetLab* [5], which has pioneered global research networks; *MoteLab* [6], which has pioneered sensor network research and *Amazon Elastic Compute Cloud (EC2)*. None of the aforementioned efforts focused on smartphones and thus those testbeds had fundamentally different architectures and desiderata. SmartLab's current hardware consists of over 40 Android devices that are connected through a variety of means (i.e., *wired, wireless* and *virtual*) to our *private cloud (datacenter)*.

Through an intuitive web-based interface, users can upload and install Android executables on a number of devices concurrently, capture their screen, transfer files, issue UNIX shell commands, perform mockup experiments by "feeding" the devices with GPS/sensor readings from big data repositories and many more exciting features that will be demonstrated during the conference.

In the context of smartphones, a *mockup* refers to the process of extending or "feeding" an AVD's or ARD's particular sensor or GPS receiver with custom values as well as supporting the addition of sensors that may not exist in the hardware of a particular ARD (e.g., NFC). In order to support both GPS and other sensor mockups (e.g., *accelerometer*) in SmartLab on both ARDs and AVDs, we opted for a custom module, coined

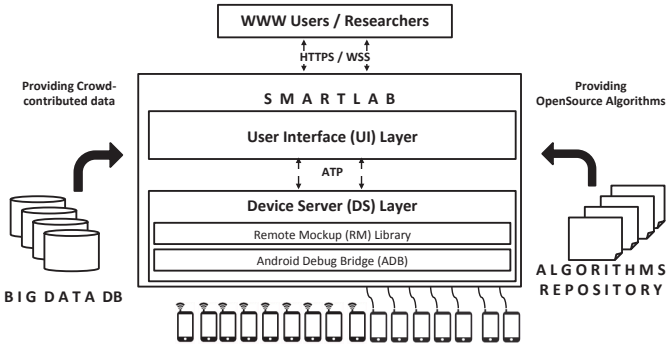

Fig. 2. The major components of the extended SmartLab Architecture involved in carrying out big data experiments.

Fig. 3. The SmartLab User Interface provides a set of tools that facilitates efficient and effective experimentation on smartphone devices.

the *Remote Mockup (RM) Library* that we plan to demonstrate during the conference. Figure 1, shows an example scenario where a user feeds nine reserved smartphones with sensor data input (i.e., accelerometer, orientation, gyroscope).

II. SmartLab Testbed Architecture

In this section, we summarize the main architectural components of the SmartLab testbed and show how these can facilitate efficient and effective experiments that utilize different algorithms, big data repositories and heterogeneous devices. We start out by briefly presenting a high level view of SmartLab's infrastructure: i) the Hardware layer; ii) the Device Server (DS) layer and administrative tools; and iii) the User Interface (UI) and Data Layers. We then move on to the specialized components that facilitate big data experiments: iv) the Algorithms repository; v) the Big Data Repository; and finally vi) the Remote Mockup (RM) Library. Figure 2 illustrates a high-level view of SmartLab's architecture and highlights the required components for big data experimentation.

A. Hardware Layer and Supported Connection Modalities

SmartLab consists of several Android Real Devices (ARDs) and Android Virtual Devices (AVDs), constructed using tools in the Android SDK (e.g., android create avd, mksdcard). Additionally, it supports a variety of connection modalities: most of the devices are directly connected to our datacenter in ARD-Local mode, utilizing USB hubs, but more smartphones are also connected using the ARD-Remote mode (i.e., *WiFi/3G*). This mode is particularly promising for scaling the testbed outside our department (e.g., ARD-Internet mode, where latencies span beyond 100ms.)

B. Device Server (DS) Layer and Administrative Tools

The Device Server (DS) is a complete Linux OS image having the SmartLab subsystems and ADB installed. Each DS is also equipped with a local web server, which is responsible to host the administrative tools required for maintenance purposes similarly to routers and printers. More specifically, SmartLab supports a variety of administrative tools: i) A "wipe" tool, which is able to simultaneously "factory reset" a number of ARD-Local or AVD devices; ii) A "backup" tool, which is able to simultaneously backup all settings, applications and files from an ARD-Local or AVD device and store them in a distributed file system; and iii) A "restore" tool, which is able to restore all settings, applications and files from a backup file (.ab) to a number of supported target devices simultaneously.

C. User Interface (UI) and Data Layers

SmartLab implements several modes of user interaction with connected devices using either *websocket-based interactions*, for high-rate utilities, or *AJAX-based interactions* for low-rate utilities. Figure 3 presents a variety of user interaction modes available through the intuitive user interface. In particular, SmartLab supports: i) *Remote Control Terminals (RCT)*, a websocket-based remote screen terminal that mimics touchscreen clicks and gestures; ii) *Remote Shells (RS)*, a websocket-based shell enabling a wide variety of UNIX commands issued to the Android Linux kernels of allocated devices; iii) *Remote File Management (RFM)*, an AJAX-based terminal that allows users to push and pull files to the devices; iv) *Remote Mockup (RM)*, a websocket-based remote mockup tool that allows to record sensor readings and GPS instances from a device and enables the replication of the pre-recorded session to a different device or group of devices; v) *Remote Debug Tools (RDT)*, a websocket-based debugging extension to the information available through the Android Debug Bridge (ADB); and vi) *Data Manager (Data)*, an AJAX-based file manager that allows users to upload and store big files to their home directory or download experimental results to their personal devices for further analysis.

D. Algorithms Repository

The Algorithm Repository stores a variety of open source and in-house developed algorithms (e.g., localization, crowdsourcing, p2p) for smartphone devices. The majority of these algorithms are packaged as stand-alone libraries (i.e., jar files) that can be used in the context of any experiment conducted using SmartLab. Documentation for each library is provided by its developer and located within the library file.

In the context of this demonstration, the Algorithm Repository will provide the Indoor Positioning algorithms[2]. SmartLab will facilitate easy integration of the data in a time-conserving deployment manner throughout the demonstration.

[2]Available under "Code" tab at http://dmsl.cs.ucy.ac.cy

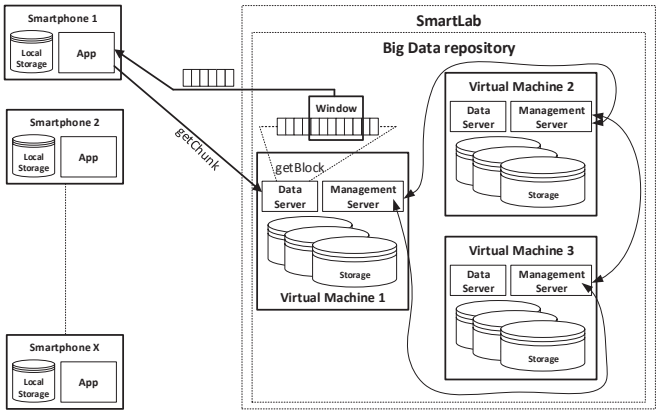

Fig. 4. **Big Data Repository Deployment Architecture**: SmartLab's big data repository is hosted by a coordinator couchbase management server virtual machine and two couchbase data servers virtual machines. It supports smartphone app query requests (`getChunk`) for data between two time instances via the `getBlock` view.

```
{
  'phone_id': 'SH0APPL00803'
  'user_id' : 'smartlab'
  [
   {
   't' : '1391198355678',
   's' : 'proximity',
   'v' : 0
    },
    {
   't' : '1391198355678',
   's' : 'accelerometer',
   'v' : ['9','1.23','4.65']
   },
   {
   't' : '1391198355678',
   's' : 'rotation rate',
   'v' : ['0','0','0']
   },
   {
   't' : '1391198355678',
   's' : 'orientation',
   'v' : ['31','6','60']
   },
   {
   't' : '1391198355678',
   's' : 'rotation vector',
   'v' : 0
    },
                          {
                        't' : '1391198355678',
                        's' : 'light',
                        'v' : 0
                         },
                         {
                         't' : '1391198355678',
                         's' : 'gyroscope',
                         'v' : ['0','0','0']
                         },
                         {
                         't' : '1391198355680',
                         's' : 'gravity',
                         'v' : ['0','0','0']
                         },
                         {
                         't' : '1391198355678',
                         's' : 'magnetic field',
                         'v' : ['-42.76','27.27',
                                '-8.58']
                         },
                         {
                         't' : '1391198355678',
                         's' : 'temperature',
                         'v' : 0
                          },
                        ]
                      }
  ]
}
```

Fig. 5. **Sensor/GPS Mockup (RM): (left, center)** A data trace of various sensor measurements encoded in JSON. The given file can be loaded to ARDs and AVDs through this subsystem; **(right)** An application built with SLSensorManager using the measurements.

E. Big Data Repository

SmartLab employs a unified big data repository infrastructure in order to maintain multiple databases or files that can be utilized in experiments on smartphone devices. The repository employs mechanisms that promote easier experimentation both at the cluster-level as well as the smartphone-level. The big data repository architecture is illustrated in Figure 4.

The data repository is currently located for distribution over a closed departmental network and has been used for storing data collected from research experiments (e.g., collecting own WiFi RSS data on campus [7], sensor data utilized by the Remote Mockup Library described next in Subsection II-F).

Generally, the given store can be utilized to store billions of sensor readings stored in a document-oriented format that allow a researcher to test an algorithm or application using tens or hundreds of smartphone devices using automated scripts, similarly to [8], but with more extensive data traces. Since each sensor recording might store millions of modeled entries without complex relational constraints, we adopted a NoSQL database that provides simplicity of design, horizontal scaling and better control of availability. We decided to use Couchbase 2.1.1 Community edition as our NoSQL database since it provides all aforementioned advantages over a relational database and it is able to accommodate unstructured JSON objects. Another advantage of Couchbase is the already built-in object-level cache, coined Memcache, which provides the ability to store and serve most frequent and recent queries immediately from the main memory (RAM) without the need of retrieving the results from the disk.

F. Remote Mockup (RM) Library

A mockup provides part of a system's functionality enabling testing of a design. As mentioned before, in the context of Android, mockups refer to the process of extending an AVD's or ARD's particular sensor or GPS with custom values. Additionally, one important benefit of mockups is that these can support the addition of sensors that may not exist in the hardware of a particular ARD (e.g., NFC, WiFi Direct).

In order to support both GPS and other sensor mockups in SmartLab (e.g., *accelerometer, compass, orientation, temperature, light, proximity, pressure, gravity, linear acceleration, rotation vector and gyroscope sensors*), we opted for a custom module coined the Remote Mockup (RM) Library. Our RM library establishes a socket server on *DS* feeding devices with sensor or GPS readings encoded in JSON format and stores them in a NoSQL repository. A sample of a constructed JSON object is depicted in Figure 5. As this functionality is completely outside the ADB interaction stream, we were required to provide each application with a custom library, coined `SLSensorManager.jar`.

The RM library can be embedded to any android application enabling interaction with the SmartLab GPS/Sensor subsystem running on *DS*. In fact, our library has precisely the same interface with the Android SDK SensorManager and consequently a user can override Android's default behavior as shown next.

- Request Internet permission in the App Manifest:

```
<uses-permission android:name=
      "android.permission.INTERNET"/>
```

- Instead of invoking the default SensorManager use:

```
mSensorManager = SLSensorManager.
      getSystemService(this, SENSOR_SERVICE);
```

- Connect to the SmartLab's Remote Mockup using:

```
mSensorManager.connectSimulator();
```

With Android Tools r18 and Android 4.0, developers have the opportunity to redirect real sensor measurements, produced by the ARDs, to the AVDs for further processing. It is important to mention that this functionality is the reverse of what we are offering. In our case, we want to be able to redirect data from a NoSQL repository to an ARD, such that a given experiment on ARDs or AVDs uses a data stream to drive its sensors.

III. Managing Big Data Experimentation on Smartphones

Smartphones are not able to accommodate large amounts of data because of hardware/software limitations. For example, an Android application limits the amount of main memory an application can use between 16MB (e.g., HTC Desire) to 48 MB (e.g., Nexus 7). This is very limiting as nowadays smartphones can produce enormous amount of data each day; if a developer desires to log the measurements of 8 sensors for a single day every 100ms, this will produce approximately 993 MB of data (i.e., 10 readings/sec $\times$ 1206 Bytes $\times$ 60 sec $\times$ 60 min $\times$ 24 hours). However, the real problem lies in the ability to replay an existing log trace to another smartphone for experimental purposes.

Assume that we would like to perform an experiment that stores a sensor reading every millisecond and our Android device can only store 16 MB of data. This means that we can replay only about 14 seconds (13911ms) of the initial recording at once. Even if the experiments allow to sacrifice accuracy (e.g., record an instance every 100ms instead of every 1ms), this would again limit our playback time to 1400s (i.e., 23 minutes).

Furthermore, the authors in [9] show that utilizing flash storage severely hampers application performance between 100% to 300% and in some extreme cases even by 2000%. Consequently, utilizing local storage for storing the aforementioned sensor readings and then retrieving them for application usage should be avoided as it can lower overall application performance, consume more energy and negatively affect experiment exact repetition.

In order to overcome the aforementioned limitation and enable the repeatability of ideally "unlimited" sizes of recordings, we have developed a "sliding window" mechanism over the sensor data stream. This mechanism enables requesting only chunks of data from the datastore using a variety of in-house developed views (e.g. `getBlock`, `getChunk` and more). Figure 4 presents a simplified version of the implemented technique. Our previous work [10] provides a more detailed analysis and a set of experiments is presented in order to illustrate how SmartLab can facilitate, automate and simplify the process of big data experimentation on smartphones.

IV. Demonstration Scenario

Interactive: We will start our demonstration out by overviewing the main components of SmartLab. We shall then present the complete interaction workflow, i.e., allocate devices, transfer files to remotely connected devices and initiate/control demonstration programs utilizing our internally developed modes of user interaction (RFM, RCT, RDT, RS and RM). More specifically, we plan to demonstrate how somebody can test both widely known applications available through the Android market and interesting applications that we have developed in-house for crowdsourced trajectory matching [4], crowdmessaging applications, fine-grained indoor localization, peer-to-peer search [3] and others.

Trace-driven: We will provide pre-recorded sensory readings and a prototype application in order to demonstrate the Remote Mockup (RM) library. Moreover, we will provide a prototype application with recording functionality in order to demonstrate how a user is able to record sensory readings through SmartLab. The trace-driven demonstration of SmartLab will utilize a pre-configured Couchbase bucket hosted by two Couchbase servers with 512MB of Memcached each (i.e., 1GB in total). The two servers can accommodate up to 6TBs of data on top of our infrastructure that encompasses over 16TB of RAID-5 / SSD storage. The data store uses 1206 bytes for storing one instance of 8 sensors readings in JSON format along with a UNIX timestamp and a username associated to the SmartLab account of the user who initiated the recording, similar to Figure 5. Finally, we will present a complete tutorial on how the RM library can be integrated into an existing Android application and mock pre-recorded sensory readings to the Android application, similar to Figure 1.

Acknowledgments

This work was financially supported by the last author's startup grant, funded by the University of Cyprus. It has also been supported by MTN Cyprus, EU's COST Action IC903 and IC1304, EU's FP7 MODAP project and EU's FP7 Planetdata NoE.

References

[1] Georgios Chatzimiloudis, Andreas Konstantinidis, Christos Laoudias and Demetris Zeinalipour-Yazti, "Crowdsourcing with Smartphones." Internet Computing, IEEE, 36-44, Sept.-Oct. 2012.

[2] Georgios Larkou, Constantinos Costa, Panayiotis G. Andreou, Andreas Konstantinidis and Demetrios Zeinalipour-Yazti. *"Managing Smartphone Testbeds with Smartlab"*, In Proc. of the 27th Intl. Conference on Large Installation System Administration (*LISA'13)*, USENIX Association, 115-132, 2013.

[3] Andreas Konstantinidis, Demetrios Zeinalipour-Yazti, Panayiotis G. Andreou, Panos K. Chrysanthis, and George Samaras. *"Intelligent Search in Social Communities of Smartphone Users"*, In Distributed and Parallel Databases, Springer US, 115-149, 2013.

[4] Demetrios Zeinalipour-Yazti, Christos Laoudias, Costantinos Costa, Michalis Vlachos, Maria I. Andreou, and Dimitrios Gunopulos. *"Crowdsourced Trajectory Similarity with Smartphones"*, IEEE Trans. on Knowl. and Data Eng. 25, 6 (June 2013), 1240-1253, 2013.

[5] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. *"A Blueprint for Introducing Disruptive Technology into the Internet"*, A blueprint for introducing disruptive technology into the Internet. SIGCOMM Comput. Commun. Rev. 33, 1 (January 2003), 59-64, 2003.

[6] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. *"MoteLab: a wireless sensor network testbed"*, In Proc. of the 4th Intl. Symposium on Information Processing in Sensor Networks, *IPSN'05*. IEEE Press, Article 68, 2005.

[7] Christos Laoudias, George Constantinou, Marios Constantinides, Silouanos Nicolaou, Demetrios Zeinalipour-Yazti, and Christos G. Panayiotou. *"The Airplace Indoor Positioning Platform for Android Smartphones"*, In Proceedings of the 13th IEEE International Conference on Mobile Data Management (*MDM'12*), IEEE Computer Society, 312-315, 2012.

[8] Tim Verry. *"MegaDroid simulates network of 300,000 Android smartphones"*, Extremetech.com, Oct 3, 2012. `http://goo.gl/jMaS8`.

[9] Hyojun Kim, Nitin Agrawal, and Cristian Ungureanu. *"Revisiting storage for smartphones"*, In Proceedings of the 10th USENIX conference on File and Storage Technologies (*FAST'12*). USENIX Association, Berkeley, CA, USA, 17-31, 2012.

[10] Georgios Larkou, Marios Mintzis, Panayiotis G. Andreou, Andreas Konstantinides, Demetrios Zeinalipour-Yazti. *"Managing big data experiments on smartphones"*, In Distributed and Parallel Databases, Springer US, 1-32, 2014.