Extending *TestelDroid* to support remote control and large-scale testing in mobile networks

Almudena Díaz-Zayas, Álvaro M. Recio-Perez, Cesar A. García Pérez, Pedro Merino
University of Málaga, Andalucía Tech, Málaga, Spain
http://morse.uma.es
[almudiaz,amrecio,garciacesaraugusto,pedro]@lcc.uma.es

ABSTRACT

This paper presents the extensions carried out in the Android measurement and monitoring tool TestelDroid, in order to support remote control and large-scale experimentation. The extensions includes support for Standard Commands for Programmable Instruments (SCPI), cOntrol and Management Framework (OMF) and OMF Measurement Library (OML). SCPI is the most widespread interface for measurement equipment control in many areas, for example, electronics or telecommunications. On the other hand, the support of technologies like OMF and OML provides powerful orchestration framework languages which reduces the time required for defining experiments.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Systems and Software; K.6 [**Management of computing and information systems**]: Software Management

Keywords

Mobile device, large-scale experimentation, remote control, SCPI, OMF, OML

1. INTRODUCTION

Experimentally driven research in mobile networks demands a high scale solution to control the execution of mobile devices. The upcoming network technologies, as well as the current increase of mobile subscribers, including machine traffic, will be significally improved by increasing the size of the research experiments on mobile networks.

Current Mobile Device Management standards[1] are complex solutions focused on functionalities such as software management, diagnostics and monitoring, all of them oriented to mobile operators. Moreover they lack of a fine control of the applications running on the device. In this paper, we introduce a simple solution to remote control of Android devices which enables extensive experimentation in the field and laboratory testing. Our approach is based on the use of SCPI commands to control the configuration of monitoring functionalities provided by TestelDroid [2], a measurement and monitoring tool for Android devices, and the deployment of OMF, an experimentation framework developed in the Global Environment for Network Innovations (GENI)[1] and Future Internet Research and Experimentation (FIRE) [2] initiatives. This framework is used to coordinate the configuration and execution of TestelDroid and the rest of applications running at the device. This work is a contribution of the UMA team to the European Project Fed4FIRE, which addressed the federation of testbeds across Europe in order to allow for large-scale experimentation with an homogeneous interface for researchers.

2. SCPI COMMANDS

The Standard Commands for Programmable Instruments (SCPI) was defined as the IEEE 488.2 specification [3] to standardize the control of programmable instruments through a serial interface. Originally, this standard defined all the levels in the interface, including physical connectors and electrical interfaces and the message exchange control. The definition of the SCPI messages (commands and responses) includes the format for messages and the state machine of the protocol. SCPI deployment was led by the SCPI Consortium until its integration into the Interchangeable Virtual Instruments Foundation (IVI) in 2002.

SCPI-enabled devices reduce the complexity of integrating measurement or control equipment in a testing laboratory or in an industrial scenario by the use of textual commands. The standardized way of control and measurement can be easily translated between devices by just changing the SCPI dialect that a particular system understands, maintaining existing algorithms already in use. This is particularly useful for the equipment vendors that can use their specialized teams to enhance the control software for new versions of the hardware. However, the direct use of this interface is still complex for many final users interested in using it for different purposes apart from the commercial one provided by vendors. Final users need a more user-friendly interface to automate some tasks on the devices.

The SCPI interface implemented compromise the functionalities defined in Table 1.

[1]www.geni.net
[2]http://www.ict-fire.eu/

VALUETOOLS 2015, December 14-16, Berlin, Germany
Copyright © 2016 ICST
DOI 10.4108/eai.14-12-2015.2262658

*IDN?	Returns a string that identifies the app.
SET:NET:REST	Resets the network by switching airplane mode on and off.
SET:MEAS:STA?:	Returns "true" if there is an ongoing measurement campaign, "false" otherwise.
SET:MEAS:STA	Starts a new measurement campaign.
SET:MEAS:STOP?:	Returns "true" if there is no ongoing measurement campaign, "false" otherwise.
SET:MEAS:STOP	Stops the current measurement campaign.
SET:MEAS:CONF:NET:EN?:	Returns "true" if TestelDroid is configured to capture network statistics, "false" otherwise.
SET:MEAS:CONF:NET:EN	Request TestelDroid to capture network statistics.
SET:MEAS:CONF:TRAF:EN?	Returns "true" if TestelDroid is configured to capture network traffic, "false" otherwise.
SET:MEAS:CONF:TRAF:EN	If called with "true", request TestelDroid to capture network traffic; if called with "false", disable network traffic capture.
SET:MEAS:CONF:GPS:EN?	Returns "true" if TestelDroid is configured to capture GPS statistics, "false" otherwise.
SET:MEAS:CONF:GPS:EN	If called with "true", request TestelDroid to capture GPS statistics; if called with "false", disable GPS statistics;
SET:MEAS:CONF:NEIGH:EN?	Returns "true" if TestelDroid is configured to capture neighbor cells statistics, "false" otherwise.
SET:MEAS:CONF:NEIGH:EN	If called with "true", request TestelDroid to capture neighbor cells statistics; if called with "false", disable neighbor cells statistics.
SET:MEAS:CONF:PROF:SCEN?	Returns the current profile scenario.
SET:MEAS:CONF:PROF:SCEN	Sets the capture profile scenario to the value passed as argument ("vehicular", "static", "pedestrian" or "high-speed").
SET:MEAS:CONF:PROF:TECH?	Returns the current profile technology.
SET:MEAS:CONF:PROF:TECH	Sets the capture profile technology to the value passed as argument ("GSM", "HSPA", "LTE", "UMTS" or "WIFI").
SET:MEAS:CONF:PROF:CONF?	Returns the current profile configuration.
SET:MEAS:CONF:PROF:CONF	Sets the capture profile configuration to the value passed as argument ("Fixed-Fixed", "Mobile-Mobile" or "Mobile-Fixed").
SET:MEAS:CONF:PROF:SUBID?	Returns the current profile SubID.
SET:MEAS:CONF:PROF:SUBID	Sets the capture profile SubID to the value passed as argument.
SET:MEAS:CONF:PROF:PEERID?	Returns the current profile PeerID.
SET:MEAS:CONF:PROF:PEERID	Sets the capture profile PeerID to the value passed as argument ("1" or "2").
SET:MEAS:CONF:PROF:COMMENTS?	Returns the current profile comments.
SET:MEAS:CONF:PROF:COMMENTS:	Sets the capture profile comments to the value passed as argument.
RET:MEAS:BATTERY?	Returns the captured battery statistics.
RET:MEAS:NEIGH?	Returns the captured neighbor cells statistics.
RET:MEAS:PROF?	Returns the captured profile statistics.
RET:MEAS:TRAFFIC?	Returns the captured network traffic.

Table 1: SCPI interface provides by TestelDroid

SCPI commands can be used to configure TestelDroid. However, their usage in conjunction with OMF framework offers a more powerful solution to orchestrate and control extensive experimentation in a repeatable and programmatic way.

3. GENI/FIRE EXPERIMENTATION TECHNOLOGY

GENI and FIRE are two initiatives for creating a research environment, fostering innovations in networking technologies, the former originating in the United States and the latter in the European Union. Interestingly, both projects are based on the same idea: experimental evaluation is a must in order to produce meaningful research that can be applied to the real world. Thus, the concept of experimental testbeds is central to the implementation of GENI and FIRE. It is worth noting that although GENI and FIRE started as two independent initiatives, Fed4FIRE, a project which is part of FIRE, has adopted GENI technologies, thus making them compatible with each other. The rest of the paper will treat the concepts pertaining to these projects as interchangeable.

Experimentally driven research demands the ability to perform experiments in an easy and repeatable manner. Testbeds have adopted OMF and OML as the solutions for providing experiment control and measurement, and experiment management, respectively. An experiment is defined as a file written in the OMF Experiment Description Language (OEDL). An OEDL file declares the resources that the experiment will use, the events to which it will react, and what actions to perform in response to those events.

The Experiment Controller (EC) interprets OEDL scripts and coordinates the execution of the experiment. Each testbed resource is managed by a Resource Controller (RC), which may be hosted on its own computer or on the resource itself. The RCs and the EC exchange control information using the Federated Resource Control Protocol (FRCP), which may be transported over the Extensible Messaging Presence Protocol (XMPP) or the Advanced Message Queuing Protocol (AMQP).

The OML server collects and stores measurements from the experiment. Each instrument, or an RC on behalf of an instrument, can send measurement data using the OML client library. Measurements are collected per experiment and can be queried using standard SQL tools.

Once the experiment script have been defined it is sent to the EC. The EC interacts with the Android RC to execute in the terminal actions defined in the script. During the experiment the OML collection server will collect all the measurements defined at the script.

4. REMOTE CONTROL OF ANDROID DEVICES

As introduced in the previous section, a resource controller is needed to control, via OMF, resources available at the experiment. The University of Thesaly has implemented a resource controller for android devices [4].

Listing 1: TestelDroid configuration through an OEDL script

```
#TestelDroid configuration
after 10.seconds do
    group("Android").exec ('su -c am
        broadcast -a com.ad.testel.COMMAND
         -e com.ad.testel.EXTRA_COMMAND
        SET:MEAS:CONF:PROF:EN -e com.ad.
        testel.EXTRA_PARAM "true"')
    group("Android").exec ('su -c am
        broadcast -a com.ad.testel.COMMAND
         -e com.ad.testel.EXTRA_COMMAND
        SET:MEAS:CONF:PROF:SCEN -e com.ad.
        testel.EXTRA_PARAM "1"')
    group("Android").exec ('su -c am
        broadcast -a com.ad.testel.COMMAND
         -e com.ad.testel.EXTRA_COMMAND
        SET:MEAS:CONF:PROF:TRAF:EN -e com.
        ad.testel.EXTRA_PARAM "true"')
    group("Android").exec ('su -c am
        broadcast -a com.ad.testel.COMMAND
         -e com.ad.testel.EXTRA_COMMAND
        SET:MEAS:CONF:PROF:COMMENTS -e com
        .ad.testel.EXTRA_PARAM "TEST"')
    group("Android").exec ('su -c am
        broadcast -a com.ad.testel.COMMAND
         -e com.ad.testel.EXTRA_COMMAND
        SET:MEAS:CONF:PROF:NET:EN -e com.
        ad.testel.EXTRA_PARAM "true"')
end

#Start monitoring
after 60.seconds do
    group("Android").exec ("su -c am
        broadcast -a com.ad.testel.COMMAND
         -e com.ad.testel.EXTRA_COMMAND
        SET:MEAS:STA")
end

#Start VoIP application
after 120.seconds do
    group("Android").exec("su -c am start
        -n com.csipsimple/com.csipsimple.
        ui.SipHome")
end

#Init a call
after 180.seconds do
    g.exec("su -c am start -a android.
        intent.action.CALL -d sip:2000")
end
```

To control mobile devices and applications running on them we use the functionality of executing ADB (Android Debug Bridge) shell commands provided by this RC. ADB provides a Unix shell that you can use to run a variety of commands on mobile device. In particular we use the activity manager (am) tool to perform system actions, such as start an activity, force-stop a process and broadcast an intent. At the same time TestelDroid have been extended to respond to a series of configuration actions issued by the delivery of intents. According to the official Android documentation an intent is a messaging object you can use to request an action from an app component. These intents contains the SCPI commands implemented by TestelDroid and the actions to start the execution of applications.

The code excerpt provided in Listing 1 is part of an experiment defined using OEDL, it shows how TestelDroid is configured and how applications and actions running on the devices are issued by the delivery of intents. You can find more details about the definition of a experiment using Android devices in [5].

Listing 2: OML measurement point definition

```
# Define the OML2 measurement point that
# TestelDroid provides. Here we
# have only one measurement point (MP)
# named 'network'.
 app.defMeasurement('network') do |m|
    m.defMetric('timestamp',:string)
    m.defMetric('networkType',:int32)
    m.defMetric('cellId',:int32)
    m.defMetric('lacId',:int32)
    m.defMetric('rssi',:int32)
    m.defMetric('networkId',:int32)
    m.defMetric('cellPsc',:string)
    m.defMetric('rsrp',:string)
    m.defMetric('snr',:string)
    m.defMetric('cqi',:string)
    m.defMetric('rsrq',:string)
  end

# Request the TestelDroid application to
# collect measurement samples from the
# 'network' measurement point (as defined
# above), and send them # to an OML2
# collection point
    app.measure('network', :samples => 1)
```

5. MEASUREMENT COLLECTION USING OML

OML provides a programming library for easy application instrumentation and a collection point, a server which stores measurements in an experiment database. The OML library [6] is available for Ruby, Python, C and Java. Java version, OML4J, can be used within an Android Application. A first approach has been implemented for TestelDroid application. Network information has been successfully sent to an OML collection server.

OML4J made the implementation relatively straightforward. OML support was embedded in the logging service of TestelDroid. When this service is started, it defines a measurement point for each logging category in TestelDroid and it tries to connect to the OML server, which is should be configure to our OML server. If the connection is successful, data is sent to the server each time it is written to the log files.

Listing 2 shows the definition in a OEDL script of a measurement point 'network' supported by TestelDroid and the request to collect these measurements.

6. CONCLUSIONS

In this paper we introduce the need for large-scale experimentally research driven in mobile network. We provide a solution based on TestelDroid as a monitoring and measurement tool and OMF/OML technology as the experiment coordination framework. The proposed architecture is in-line with current tools promoted by Fed4Fire project and FIRE community: OMF, OML and OEDL. Future work will include the use of jFed tool [7] to design the topology design and the reservation of the resources.

Currently the PerformLTE testbed [8] offers the access to mobile devices connected to commercial mobile networks available at Spain. Reference experiments can be downloaded at `http://www.lcc.uma.es/~pedro/mobile/Software/testeldroid.html`

7. ACKNOWLEDGMENTS

This work has been funded by the Government of Andalusia under grant P11-TIC-7659, by the Spanish Ministry of Economy and Competitiveness under project TIN2012-35669, European Regional Development Fund (EDRF) from the European Commission and with the support of the Fed4FIRE-project (Federation for FIRE), an Integrated project receiving funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 318389.

8. REFERENCES

[1] OMA, "OMA Device Management," Tech. Rep. V2.0, Open Mobile Alliance (UMA), Jan. 2015.

[2] A. Alvarez, A. Diaz, P. Merino, and F. Rivas, "Field measurements of mobile services with android smartphones," in *Consumer Communications and Networking Conference (CCNC), 2012 IEEE*, pp. 105–109, Jan 2012.

[3] IEEE, "IEEE standard serial interface for programmable instrumentation," *IEEE Std 1174-2000*, pp. i–30, 2001.

[4] D. Stavropoulos, G. Kazdaridis, N. Makris, H. Niavis, I. Igoumenos, T. Korakis, and L. Tassiulas, "Enabling experimentation in mobile sensing scenarios through 4g networks: The nitos approach," in *Networks and Communications (EuCNC), 2015 European Conference on*, pp. 502–506, June 2015.

[5] "Running OMF 6 on the android platform." `https://omf.mytestbed.net/projects/omf/wiki/Android`.

[6] "OML library." `http://oml.mytestbed.net/doc/oml/latest/doxygen/files.html`.

[7] "jFed is a java-based framework for testbed federation." `http://jfed.iminds.be/`.

[8] C. A. Garcia-Perez, A. M. Recio-Perez, A. Diaz-Zayas, and P. Merino-Gomez, "PerformLTE: a testbed for LTE testing in the Future Internet," in *Wired/Wireless Internet Communications, WWIC 2015, LNCS 9071*, pp. 313–326, 2015.