Lightweight Device Task Actuation Framework as IoT Test Platform

Dhiman Chattopadhyay(✉), Abinash Samantaray, and Hari Raghav

Innovation Lab, Tata Consultancy Services, Kolkata, India
{dhiman.chattopadhyay,abinash.samantaray,hariraghav.1}@tcs.com

Abstract. Popular test automation frameworks target the enterprise application testing but there is scarcity of test automation framework for device applications, especially for IoT domain. IoT testing paradigm throws a new set of challenges involving device integration, protocol adapters, task actuation, data integrity, security and non functional requirements. In this paper, we propose a scalable, lightweight device task actuation framework for IoT testing based on TCS Connected Universe Platform Device Management enabler. This framework can execute test suite on multiple remote devices spread across geographies and then show the results on the IoT tester's screen. Moreover it has the ability to gather runtime device statistics during test execution, thus can do dynamic health check for IoT devices deployed on field.

Keywords: TCUP · DM · DTAF · ITP · LWM2M · CoAP · REST · IoT

1 Introduction

The important consideration in case of a software testing are safety, reliability, resilience, availability and security. Legacy Test Automation frameworks [1,2] are web based tool to execute test suite without manual intervention and monitoring. There are tools [3] to automate enterprise application testing where applications deal with predominantly human generated data. Such tools deal with UI testing and as well as functional testing of enterprise application with predefined set of inputs and arrive at pass fail decision in comparison with reference results. With the arrival of era of Internet of Things (IoT) devices are set to take the driver seat in terms of data generation, acquisition, transfer and task actuation. IoT solution testing involves validation of functionality, security, actuation and benchmarking for testing the reliability, security, stability and performance of IoT solution. The IoT devices vary in hardware type ranging from constrained embedded microcontrollers like Arduino and mbed platform to more resourceful gateways like RaspberryPi, Intel Galileo, Edison, Beagle Bone etc. and even smart-phones can act as a gateway. But the commonality among the edge devices is the ability to acquire, communicate and compute. As the importance of device application is paramount in IoT application it has become necessary to rigorously test the device applications before field deployment. There are popular

B. Mandler et al. (Eds.): IoT 360° 2015, Part II, LNICST 170, pp. 20–27, 2016.
DOI: 10.1007/978-3-319-47075-7_3

commercial and open source test automation tools like Winrunner, Selenium widely used in web application test automation. But there is a void in the availability of test automation framework for IoT applications dealing with machine generated data and task actuation. There are popular mobile test automation framework like Sikuli, Robot, Monkeyrunner for mobile app testing but those are not portable on constrained devices for IoT application testing. The reason of unavailability of such test automation support in IoT domain is due to resource constraints of the device under test (DUT) as aforesaid tools use resource hungry protocols requiring large amount of resources like CPU, memory, power and bandwidth. Presently the testing of device applications is done at a central site before field deployment. Engineers execute the test suite and collect the results from tests running on individual devices. Manual testing of device application involves tedious manual intervention and constant monitoring.

2 TCUP Overview

TCS Connected Universe Platform (TCUP) [4,5] is an IoT PAAS offering that includes Device Management (DM), Message Router service, Sensor observation service (SOS), Data Analytics service and Complex Event Processing service. TCUP is a cloud based multitenant Platform-as-a-Service (PaaS) offering that makes it easy to develop, deploy and administer M2M or IoT applications. TCUP consists of a set of RESTful modules exposing web services, device agents and web portal that are specifically designed for application developers to create highly scalable and intelligent analytics driven applications that make use of sensors and devices. TCUP can be hosted on private cloud like Openstack or public cloud like AWS or Azure. Figure 1 depicts a high level overview of TCUP. TCUP also serves as a horizontal IoT application management platform where vertical domain specific services can be hosted with which corresponding device agents can interact, generated events can be stored, analyzed and finally some action can be triggered based on the application decision logic.

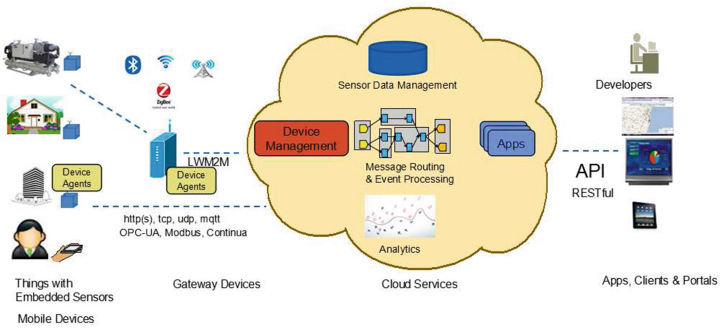


Fig. 1. TCUP modular overview

3 Device Management Concept

TCUP platform has a Device Management (DM) module which allows remote monitoring of devices and sensors, monitor health and connectivity status of devices, observe devices and their resources send commands to devices. TCUP supports the OMA LightweightM2M(LWM2M) standard [6] which uses underlying protocol CoAP [7]. The choice of CoAP over HTTP in IoT is gaining impetus as it is less resource hungry in terms of CPU load, memory footprint and bandwidth usage and energy consumption [8]. LWM2M follows Client-Server architecture where the Device hosts the LWM2M client and the DM server hosts the LWM2M server. LWM2M uses the Constrained Application Protocol (CoAP) with UDP bindings for transport. Each property of a device is modeled as a CoAP resources and similar resources are clubbed under one LWM2M objects for logical grouping. For e.g. location related resources like latitude, longitude and altitude resources are grouped together under location object.Multiple DM clients' resources form a hierarchic tree at DM server side and DM Server can access each device through CoAP resources URIs. DM server maintains the mapping between device endpoints and their IP along with their corresponding resource sub-tree structure and IP gets updated during periodic registration update by devices. LWM2M protocol stack with CoAP-HTTP bidirectional proxy forms the core engine of DM module. The DM service layer atop provides a RESTful interface which can be consumed by DM portal or other third party application. Following Fig. 2 depicts an high level architecture of DM service module. The notifications on any observable resources are automatically posted by DM agents to DM server. The observed data may be consumed by any application through Message Router (MR) module like posting the observation data to TCUP Sensor Data Management module. For scalability and high availability server clusters can be formed with UDP load-balancer for LWM2M core and HTTP load-balancer for DM service.

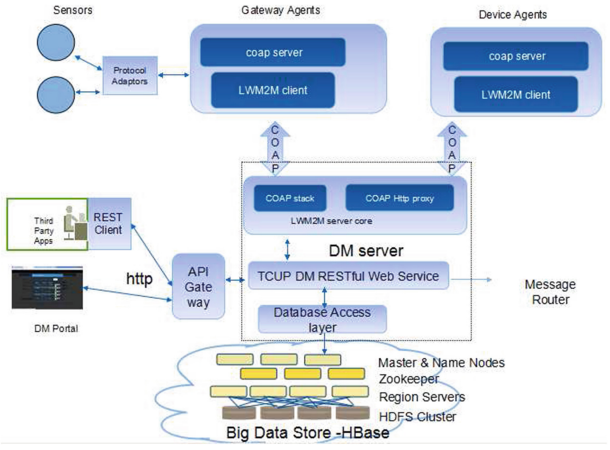


Fig. 2. TCUP DM service high level architecture

Access control list (ACL) defines permissible operation on a resource. API gateway authorise and authicates every call based API key passed with service call. Device data is stored in a NoSQL database (HBase). Every device resource is either static (like make, IMEI etc.), dynamic (like sensor value, CPU load etc.) or editable (like device description). If DM resource is observable then events for that observable resource can be subscribed by DM user through RESTful interface of DM service in order to receive automatic notification if there is any change in the resource value. This feature is useful to get automatic notification every time a sensor value changes or some events occur.

4 Device Task Actuation Framework High Level Overview

TCUP Device Task Actuation Framework (DTAF) as IoT Test Platform (ITP) is a cloud based solution on TCUP PAAS offering which facilitates the tester to remotely execute IoT test applications on multiple devices and get the results. ITP also enables the tester to view runtime device data varying CPU load or RAM usage and as well as sensor values during test execution. In case of DTAF device agent runs on DUT and works as test controller. High level overview of lightweight device test automation framework is shown in Fig. 3. Tester can launch test applications on remote DUTs from any device and get test results after test completion.

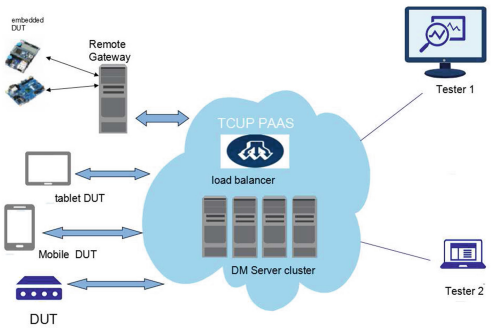


Fig. 3. Device task actuation framework overview

The device agent is modeled as a group of RESTful CoAP resources which represent static or dynamic device parameters, configurations and actions. First device needs to commission itself with TCUP DM server which is initiated by the DM agent registration with DM server, henceforth device can be managed through TCUP. Java implementation of device agent architecture is shown in the Fig. 4. DM agent includes CoAP resource class for every resource where every individual resources can have GET and PUT handlers depending on read/write

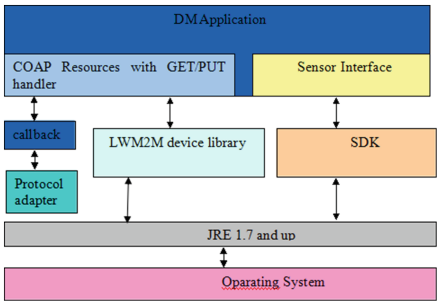


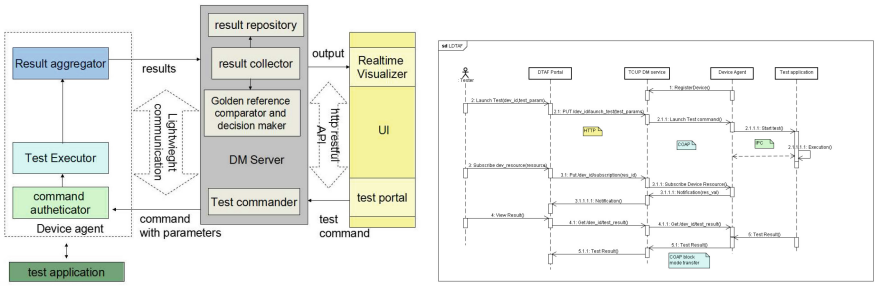
Fig. 4. Device agent highlevel architecture

access control list. Protocol adapters enable agent to communicate with sensors over various protocols. The sensory resources also invokes OS specific sensor interface API to acquire sensor data. Test launcher is defined as a CoAP resource in DM agent with CoAP GET/PUT handlers which serves test launching request coming from the DM server end.

4.1 Test Execution Process in DTAF ITP

As tester fires a test from portal the DM Server (DMS) gets test command through DM Rest API call from test portal, a consumer of DM services. DM service translates command from Http to CoAP and send that to DMC over CoAP protocol. DMC works as test controller and has a resource "launcher" with GET PUT handlers. Receiving test command launcher's PUT handler invokes a callback that autheticates the command and then fires the specific preinstalled test application on device. Then test application gets test parameters from DMC and performs execution accordingly. Now Device agent can get runtime data from test application over any inter process communication (IPC) mechanism available for DUT's OS like a domain socket in Linux or an intent broadcast receiver mechanism in Android. During test execution user can opt for subscribing runtime data from DUT through portal and can view the runtime device parameters along with sensor data on TCUP Sensor Data Explorer (SDE), a HTML5 based visualizer. DMC sends test result to DMS over CoAP and DMS does the CoAP to HTTP translation and sends to portal over HTTP. Then portal shows the result. The interaction between different modules of DTAF is explained with a Fig. 5a.

Device agent accepts commands from DMS, controls test execution, gathers dynamic device data, aggregates generated result and send back the result to DMS. If test nature requires a pass-fail decision then generated result is compared against stored golden reference either at device side or at server side. Transfer of result set at server for comparison involves more bandwidth usage whereas comparison at device side loads device computation resource. So the choice of place of decision making requires a trade off between device hardware capacity and bandwidth availability. Following sequence diagram in Fig. 5b depicts the aforesaid message flow between DTAF entities.

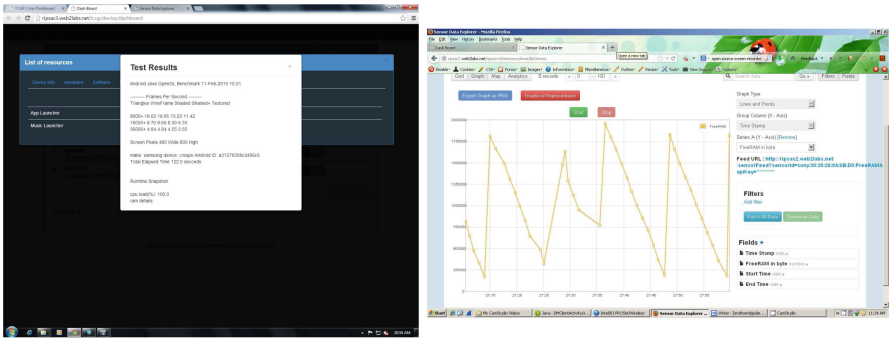


(a) ITP block diagram (b) DTAF Sequence Diagram

Fig. 5. DTAF message flow

5 Prototype Implementation and Results

Tester first runs device agent i.e. DM client (DMC) on Device under test (DUT) and registers it in TCUP. He logs into ITP portal to see list of DUTs registered by him and he selects a particular device and view the details of the selected device. As the tester clicks on the launch button against test name from action tab test starts running on the DUT. The result button and launch button will be disabled during test execution. During execution a tester can opt to see dynamic resources (like sensor)on a live graph on SDE by subscribing the resource through portal, live graph runs until tester opts out by unsubscribing the resource. Once the test is finished the test result can be seen by clicking on result button against the test. Figure 6a shows a screenshot of a test result by running a sample benchmark test on device while Fig. 6 shows live graph in TCUT DTAF ITP portal. This sample test is based on open source benchmark algorithm like Wheatstone, Dhrystone, Linpack etc.



(a) Test Result on Portal (b) Dynamic Data Live stream

Fig. 6. Screenshots from ITP prototype

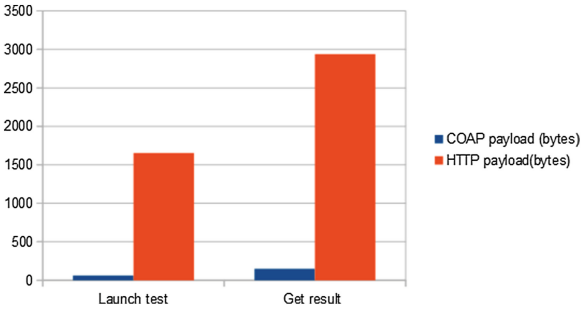


Fig. 7. Bandwidth savings

5.1 Bandwidth Usage and Performance Measurement

Less bandwidth is used by CoAP based DTAF framework against a HTTP based mobile test automation framework. Average bandwidth usage by DTAF over 100 iterations shows the improvement in bandwidth consumption in Fig. 7. Although bandwidth is getting cheaper with time, the traffic reduction continues to be an important consideration due to huge number of devices. From results it is evident that CoAP payload uses hundred times less bandwidth than HTTP to transmit the same information.

We have measured performance of our test automation framework through load testing using Apache JMeter. The performance test setup comprised of the following hardware configuration: 1. Phoenix 4.2.2 plus HBASE 0.98 cluster running on HDFS with 1 zookeper, 1 hbase master and 3 region servers, all with 1 core, 2 GB RAM Ubuntu VMs on Openstack cloud 2. DM Service, developed using Spring v4, running on Apache Tomcat 7.47 on a Intel Core i5, 4 GB RAM machine with 32 bit Windows 7. DM Server is allocated 268MB of Memory. 3. Internet Connectivity with 100 Mbps backbone for server and 3G for devices The result is given below in Table 1.

Table 1. Performance measurement: concurrent request/second

Throughput per sec	100	500	1000
Query result (GET)	45	43	42
Launch application (PUT)	87	85	83

Study of results reveals that the performance in launching test is better than querying the result. The reason is launching application involves passing on a JSON payload comprising of test name and parameters whereas result query returns a large payload which internally gets sliced into multiple chunks handled by CoAP block mode transfer protocol. Also DM's underlying database is HBase where read has more latency than write operation [9]. It is evident

from result that throughput remains nearly constant despite increasing number of concurrent request handled by DM server, so the complexity of test launching is O(1) independent of individual test application's complexity. This behaviours is expected as one TCUP DM server has capacity to handle 10000 request per second, although performance measurement with further increase in load is a subject of our future scope of work. To serve load more than 10000 request per second would require server cluster with load balancer.

6 Conclusion

DTAF based ITP uses Lightweight M2M protocol to make this framework a suitable candidate for IoT device application test automation. The runtime data collection feature during execution enables hardware behaviour test for sensor intensive gaming consoles. ITP can be utilized to run resident device health check and security verification script during device idle time. The device task actuation framework can be leveraged to design smart home or smart automative solution [10]. For e.g. this framework can be used for contextual actuation of a remote media controller to control a home entertainment device. Controlling of device side actuation poses a new security threat that generates the need of lightweight authentication and authorization of device commands which will be our new area of research.

References

1. Wang, F., Du, W.: A test automation framework based on WEB. In: 2012 IEEE/ACIS 11th International Conference on Computer and Information Science (ICIS), pp. 683–687, 30 May 2012–1 June 2012
2. Fu, L.L., Dai, J.Q., Liu, J.H.: Auto test solution for web application. Inf. Techonl. **4**(39), 23 (2010)
3. Wu, Y.: Automation testing framework for web base on web base on selenium. Inf. Techonl. **9**, 187–188 (2011)
4. Misra, P., et al.: A computing platform for development and deployment of sensor data based applications and services. Patent No. WO2013072925 A2
5. http://www.tcs.com/SiteCollectionDocuments/Brochures/Innovation-Brochure-TCS-Connected-Universe-Platform-1014-1.pdf
6. http://technical.openmobilealliance.org/Technical/technical-information/release-program/current-releases/oma-lightweightm2m-v1-0
7. Shelby, Z.: Constrained Application Protocol (CoAP) RFC6690. http://tools.ietf.org/html/rfc6690z
8. Sammarco, C., Iera, A.: Improving service management in the internet of things. MDPI Sens. **12**(9), 11888–11909 (2012)
9. http://planetcassandra.org/nosql-performance-benchmarks
10. Ghose, A., et al.: Internet of Things application development. Patent No. EP2806356 A1