CCT: Connect and Control Things

A Novel Mobile Application to Manage M2M Devices and Endpoints

Soumya Kanti Datta, Christian Bonnet, Navid Nikaein
Mobile Communication Department, EURECOM
Biot, France
Emails: {dattas, bonnet, nikaeinn}@eurecom.fr

***Abstract*—This paper presents a novel application that allows mobile clients to interact with M2M devices and endpoints in real time. The application "Connect and Control Things" (CCT) is designed to discover things, receive data from the sensors, control the actuators and generate alarms in real time. The novel capabilities of CCT are: (i) dynamic discovery of device and endpoint, (ii) real time interaction with sensors and actuators associated to M2M devices, (iii) benefit from Sensor Markup Language (SenML) representation, (iv) supporting extension of SenML capabilities for actuators and (v) learning actuators' resources and control them. The architectural design, prototypes implementation, flow of network operations and a real-life test scenario are illustrated. The prototype Android application registers higher CPU usage and power consumption due to intense network operations and parsing sensor metadata repeatedly. We have proposed several optimization techniques to reduce the CPU load, network data usage and overall power consumption. Two use cases of the application have been discussed. Finally the paper summarizes the contributions and concludes with the future research directions.**

***Index Terms*— Android; Dynamic Discovery; M2M Device and Endpoint; M2M Gateway; Mobile Application; SenML.**

I. INTRODUCTION

The development in Internet of Things (IoT) ecosystem is essentially driving the growth of Machine-to-Machine (M2M) market [1, 2]. IoT has also opened new vistas in smart metering, smart grid, smart city, automotive industry, e-Health, sensor and actuator networks [3, 4, 5]. This tremendous growth has made mobile applications for IoT ecosystems totally indispensable. For example, in smart homes, M2M devices [6] are able to detect if a light bulb is on while there is no one in the room. That is reported to the user through a mobile application and the user can send a command to switch off the bulb from the same. The most common functionalities in such mobile applications are: (i) connecting to M2M devices containing sensors and actuator as endpoints, (ii) gather periodic updates and (iii) control actuators from the mobile clients.

This paper is focused on describing a mobile application "Connect and Control Things" (CCT) that achieves the mentioned functionalities while providing novel services. The application enables real time interaction with M2M devices. Each device manages a sensor and/or an actuator which are also known as things or endpoints. One possible deployment scenario of CCT is the usage of a M2M gateway between the mobile clients and the M2M devices. The M2M devices and endpoints are connected to the gateway over IP links. The gateway is a collection of APIs implemented using RESTful web services. The sensor measurements are represented using Sensor Markup Language implementation[1]. SenML exists as a work-in-progress draft and is defined by a data model for sensor measurements along with a metadata about measurements and devices. But SenML draft addresses only sensors at the moment and its capabilities must be expanded to support actuators.

Rest of the paper is organized as follows. Section II and III provide a requirement analysis of the application and describe SenML extensions respectively. Section IV explains the architectural design and different modules of CCT including the novel services. The prototype implementation, network operations are illustrated in Section V while Section VI focuses on performance evaluation and optimization techniques. Section VII discusses two use cases that benefit from CCT. Then the paper concludes with future research directions.

II. REQUIREMENT ANALYSIS

In order to achieve the mentioned functionalities, the application is required to provide the following services:

- **Dynamic discovery:** This is one of the novel aspects of the application. The application does not have a-priori knowledge about M2M devices and endpoints. Thus it must perform a discovery of the devices and endpoints connected to the M2M gateway. M2M devices might be added to or removed from the gateway which will be reflected during the discovery and reported to the users. The operations are thus M2M gateway assisted.
- **Real time interaction:** The application must support real time interaction with both sensors and actuators. Also it must be able to raise alarms based on the local computation on sensor measurements.
- **Connecting to smart and legacy things:** The things could be smart (allows to read metadata using GET request) or legacy (non-smart). The application CCT is able to connect to both kinds of things and read metadata using GET requests over HTTP.
- **Subscribe to PUSH notification:** This is necessary to receive updates over HTTP from the M2M gateway when (i) sensor reading is changed and (ii) new endpoints are added to or removed from it.
- **SenML implementation:** SenML is designed to be very lightweight and can be parsed efficiently. Thus SenML implementation is the natural choice to carry the sensor metadata from M2M devices to the mobile

[1] http://tools.ietf.org/html/draft-jennings-senml-10

clients. The implementation is done using JavaScript Object Notion (JSON) which is more suitable for a concise things representation against an XML-based format.

- **Actuator control:** Currently SenML implementation exists only for sensors. But we have extended the capabilities of SenML so that the same JSON implementation can be used to control actuators also. This is a major innovation provided by the application and is described in details in the following section.
- **Cross platform approach:** It is necessary to develop the application for multiple mobile OS platforms to reach out to maximum potential users and reduce the time to market. Cross platform development provides immense benefit in this respect. A comparative study of most commonly used cross platform tools is provided in [9]. CCT is developed using such approach.

III. Extending SenML Capabilities

Current SenML capabilities are expanded to control actuators. The main motivations behind such extensions are: (i) to provide uniform representation of sensor and actuator metadata and (ii) use the same software implementation SenML to control actuators. We have introduced the following attributes for actuators:

- **An Interface Definition**: It is necessary to distinguish between a sensor and an actuator.
- **Name of actuator:** This is a unique name that identifies the actuator and represented by base name.
- **Type of actuator**: It defines various types of actuator e.g. temperature controlling actuator.
- **Allowed range of values**: The mobile clients must know the allowed range of values in order to control the actuators. The range could be continuous (e.g. for a motor) or discreet values (e.g. 0/1 for LED).
- **Unit**: The unit of the values.
- **Capabilities**: It signifies whether an actuator is smart or legacy endpoint. In case of a legacy actuator, another M2M device must translate the instructions to machine executable form.
- **Semantic**: It is used to associate a semantic notion of the actuator operation and can be pre-configured for the actuators.

Along with the above extensions, following two attributes are also necessary.

- **Location**: It signifies the type of actuator location and can be denoted by GPS co-ordinates, XY location or semantic location (e.g. Room 313 or Building A).
- **Destination**: It denotes the URI of the actuator and the control commands are sent to this URI from the clients.

The IPSO Alliance Framework[2] already defines generic RESTful resource representations for actuators to be used with SenML and they provide a way of switching on/off a LED actuator. But our implementation extends the SenML capabilities beyond that by adding name, type, unit, location, destination, (a semantic for operation performed) and capabilities for actuators. Current IPSO draft mentions only discreet value of allowed range for actuators but we have added support for continuous range of values. These extensions are supported by all the components in the following architecture.

IV. Architectural Design And Novel Services Implementation

The overall architecture containing the mobile clients (running CCT), M2M gateway, M2M device and endpoints are shown in Figure 1. The smart M2M devices can directly interact with the M2M gateway and exchange sensor metadata. But the legacy devices have to be connected through another intermediate gateway known as M2M sensor gateway. It is used for data aggregation from the legacy things and is configured with necessary information (unit, timestamp, name etc.) to create metadata in SenML format. M2M sensor gateway can be generalized to support different types of legacy things. The M2M gateway contains several RESTful APIs to manage the M2M devices, endpoints and their configuration resources. Each such device must register its configuration details along with those of endpoints to the gateway in order to facilitate discovery phase of CCT.

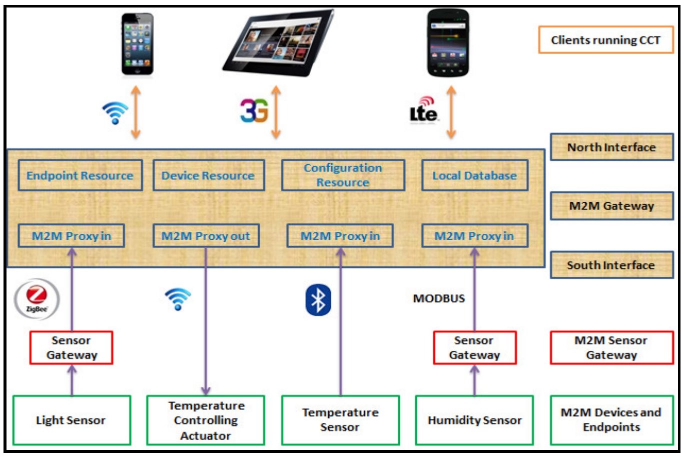


Figure 1. Overall IoT architecture.

SenML along with its extensions provide a single formalism to represent both sensor and actuator in the above architecture. HTTP is used to carry SenML metadata, send GET request and. The software architecture of CCT is depicted in Figure 2 and can be broadly categorized into four modules as described below.

A. *Device and Endpoint Discovery Module*

Upon launching, CCT establishes a connection to gateway to initiate the dynamic discovery phase which takes place in two steps. Firstly, this module sends a GET request over HTTP to retrieve a list of M2M devices. The response contains the description in terms of resource type and attributes including location, id, name, and software version of the device [7, 8].

[2] http://www.ipso-alliance.org/wp-content/media/draft-ipso-app-framework-04.pdf

Secondly, after device discovery, the user must select a M2M device to receive the list of attached endpoints. This is done using another GET request which triggers endpoint discovery. The response includes: (i) id – the resource identifier, (ii) name: human readable name, (iii) device: a link to the M2M device connecting the endpoint, (iv) location: URI of the endpoint (v) senml – the SenML metadata and more fields. It is to be noted that, for multiple M2M devices and/or endpoints, the client receives multiple responses from the M2M gateway. This drives up the amount of network operations. The discovery phase is based on CoRE Link format [7, 8] while the resource type of location is based on IPSO Alliance Framework. This phase is also crucial to learn the description of actuators using the SenML extensions.

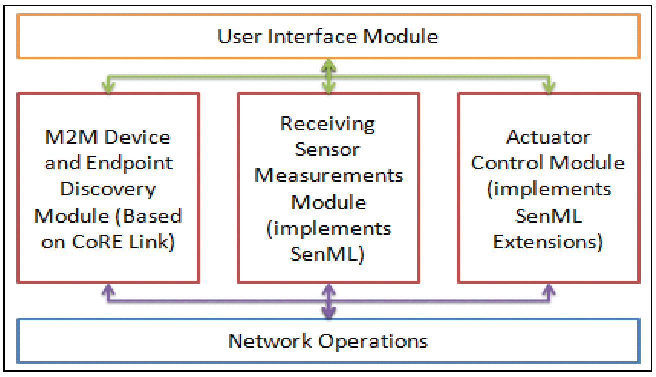

User Interface Module
M2M Device and Endpoint Discovery Module (Based on CoRE Link)
Receiving Sensor Measurements Module (implements SenML)
Actuator Control Module (implements SenML Extensions)
Network Operations

Figure 2. Software architecture of CCT.

B. *User Interface Module*

After the discovery phase, CCT displays a list of available sensors and actuators along with their attributes. The user can select a sensor to receive the measurements. To provide skeuomorphic user experience, every sensor and its values are associated with both numeric and iconic (picture) representation. An innovative aspect of the UI module is that, certain components of the user interface can be dynamically configured. For example, the value to be sent to an actuator could be implemented using a slider, input text box or checkbox with numeric values. The motivation behind this is to drive the user interface components from the M2M gateway depending on different types of endpoints.

C. *Receive Sensor Measurement Module*

This module of CCT can receive the most recent sensor metadata by pulling the same from the M2M devices via the M2M gateway. Also the application can subscribe to the receive push notification from the M2M gateway which sends automatically the updated metadata. This module can generate alarms (vibration, sound) or alerts (SMS) based on predefined rules applicable on received sensor measurements.

D. *Actuator Control Module*

During the discovery phase, CCT learns about the actuators. This module supports the implementation of SenML extensions for actuators. To control an actuator, the appropriate value is set. Then the module generates the SenML representation of the command containing name, value, unit, location of actuator. The generated metadata is then sent to the M2M gateway using POST over HTTP which forwards the metadata to the corresponding M2M device containing the actuator. The response code 204 is also received by this module and is updated to the UI module to provide a feedback.

V. Prototype Android Application

The prototype mobile application is implemented using PhoneGap 2.9.0 and JQuery Mobile 1.3.1. The cross platform approach allows building the application for multiple mobile platforms and minimizes the cost of development and time to market. The M2M gateway APIs are running in a Google App Engine (GAE) server. The application establishes a connection to the server for M2M device and endpoint discovery. But due to current specification of CoRE Link, to receive the complete list of resource types and attributes for devices and endpoints, multiple GET requests are needed. This increases the load on CPU and network operations. The sensor measurements can be obtained either by pulling or push notification. For the Android application, push notification service uses Google Cloud Messaging (GCM). The service is portrayed in Figure 3 and is explained below.

1. The Android application CCT registers the client with Google Cloud Messaging (GCM) service.
2. GCM assigns a unique ID (UID) to each client and sends the UID to the client.
3. The client then communicates the UID to the GAE. This is the client registration at the GAE for push notification.
4. In the event of a sensor measurement update, the GAE receives the sensor metadata which is sent to the GCM along with the UID. The GCM service pushes the metadata to the client corresponding to the UID.

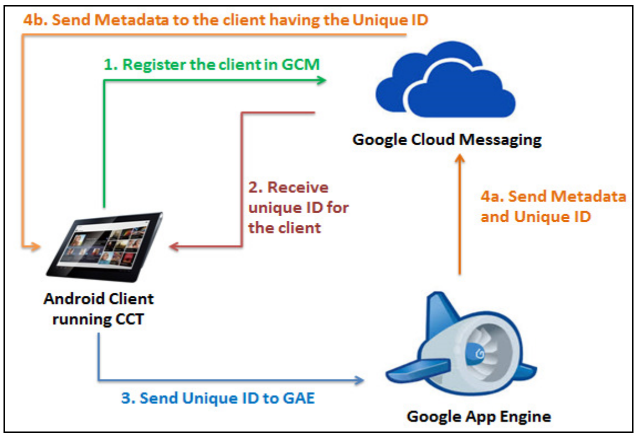
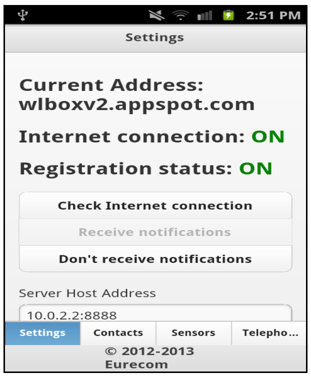

4b. Send Metadata to the client having the Unique ID
1. Register the client in GCM
Google Cloud Messaging
2. Receive unique ID for the client
4a. Send Metadata and Unique ID
Android Client running CCT
3. Send Unique ID to GAE
Google App Engine

Figure 3. Push notification architecture for Android devices.

For the iOS clients, Apple Push Notification Service has to be used instead of GCM and rest of the mechanism remains the same. When the application is launched, there is an option to register for the push notification which is denoted by "*Receive notifications*" tab in Figure 4. "*Registration status: ON*" denotes CCT is subscribed to the push notification service. The

"Current Address" is the URL of GAE server running the RESTful APIs of the M2M gateway.

Upon clicking the "*Sensors*" tab, the discovery phase begins. Figure 5 shows that only one M2M device (named T-Aix) containing a temperature sensor having value 24 degree Celsius and an actuator to control the temperature. The SenML extensions for actuator are implemented in the backend. But it is seen that to control the temperature, the desired value can be set using the slider which is automatically configured with the range of allowed values and the unit being same as the sensor. This M2M device can be deployed to a room to monitor the temperature and control it from CCT.


2:51 PM

Settings

Current Address: wlboxv2.appspot.com

Internet connection: ON

Registration status: ON

Check Internet connection

Receive notifications

Don't receive notifications

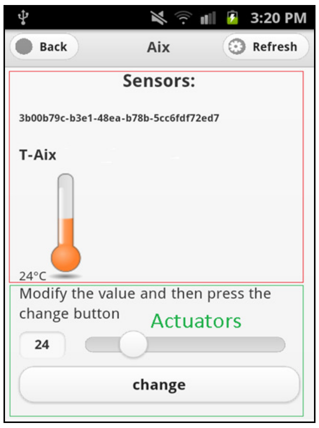
Server Host Address

10.0.2.2:8888

Settings | Contacts | Sensors | Telepho...

© 2012-2013 Eurecom

Figure 4. CCT launch screen.


3:20 PM

Back | Aix | Refresh

Sensors:

3b00b79c-b3e1-48ea-b78b-5cc6fdf72ed7

T-Aix

24°C

Modify the value and then press the change button

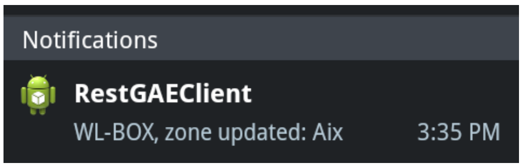
Actuators

24

change

Figure 5. M2M device with a sensor and an actuator.

To demonstrate the actuator operation, the temperature is set to 10 degree Celsius using it. CCT sends the command using SenML extensions for actuators. Once the temperature change has taken place, CCT receives a push notification and UI module of the app shows the updated temperature as portrayed in Figure 6 and 7 respectively.


Notifications

RestGAEClient

WL-BOX, zone updated: Aix 3:35 PM

Figure 6. Push notification for sensor measurement update.

It is to be noted that the intermediate temperatures (23, 22, …, 12, 11) are intelligently filtered out by the M2M gateway and is not pushed to the device. When the temperature reaches 10 degree Celsius, the corresponding metadata is pushed to the device. The intelligent filtering at the M2M gateway is done to improve user experience, avoid unnecessary network operations and reduce processing load on CCT.

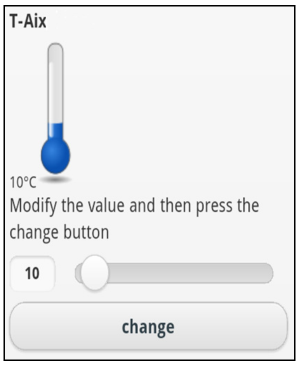


Figure 7. Setting a new temperature using the actuator.

If a new M2M device (L-Aix) with light sensor is added to the M2M gateway, CCT will receive a similar notification as shown in Figure 6. Also the UI will be updated as shown in Figure 8.

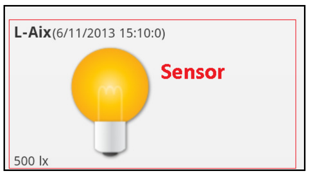


Figure 8. New M2M device with a light sensor added to M2M gateway.

The prototype application is tested in real life environment with a M2M device and two endpoints. The overall flow of operations of the test scenario is presented in Figure 9. Before initiating the discovery phase, CCT subscribes to the push notification service. Currently the prototype supports discovery phase, receiving sensor measurements using pull & push and sending instructions to control an actuator.

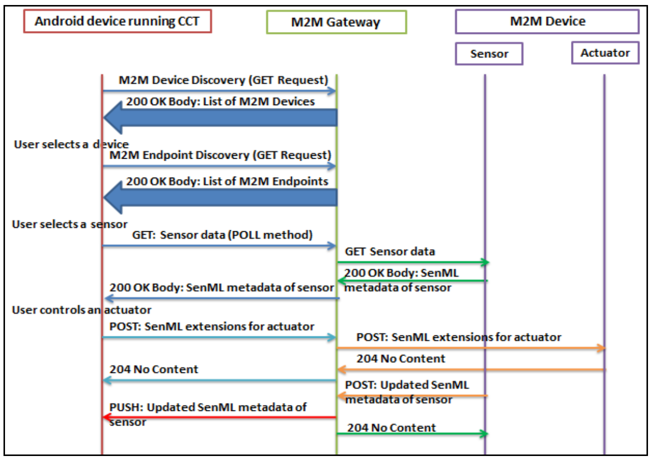

Android device running CCT
M2M Gateway
M2M Device
Sensor
Actuator
M2M Device Discovery (GET Request)
200 OK Body: List of M2M Devices
User selects a device
M2M Endpoint Discovery (GET Request)
200 OK Body: List of M2M Endpoints
User selects a sensor
GET: Sensor data (POLL method)
GET Sensor data
200 OK Body: SenML metadata of sensor
200 OK Body: SenML metadata of sensor
User controls an actuator
POST: SenML extensions for actuator
POST: SenML extensions for actuator
204 No Content
204 No Content
POST: Updated SenML metadata of sensor
PUSH: Updated SenML metadata of sensor
204 No Content

Figure 9. Flow of network operations for CCT.

VI. Performance Evaluation and Optimization Techniques

As seen from Figure 9, most of the functionalities of CCT like fetching the list of M2M devices & endpoints, sensor metadata and actuator control depend on intense network operations. Then the received responses from the gateway have to be parsed every time to update UI. This drives up the CPU load and operating frequency. As identified in [10, 11], increased network operations and high CPU load are two key factors responsible for higher power consumption in smart devices. Thus we have evaluated the performance of CCT based on the two mentioned metrics. The experiments are performed on three Android devices – (i) Archos Tablet with Android 4.0.3, (ii) Nexus S with Android 4.1.2 and (iii) Nexus 7 with Android 4.2.2.

A. CPU usage results

The CPU load is measured during four phases of operation of CCT and the results are listed in Table I. The native part of the executable is loaded at the beginning of application execution. The discovery phase registers higher usage of CPU as multiple GET requests are made and the received data are parsed multiple times before the UI can be updated. For the same reason the CPU load is higher for receiving and parsing the sensor measurements. When CCT is just displaying a list of endpoints or sensor measurements, it consumes limited CPU resource as the application is not doing any network operations and parsing.

TABLE I. CPU Usage of CCT

Android Device	**Native part loading**	**Device and endpoint discovery**	**Parsing sensor values**	**Sensor data representation in UI**
Archos Tablet	24%	32%	37%	4%
Nexus S	51%	41%	40%	8%
Nexus 7	17%	35%	36%	3%

B. Power Consumption

The power consumption results are measured using another Android application Power Tutor[3] and tabulated in Table II. Due to high amount of network operations and CPU intensive parsing, the overall power consumption is significantly high on all devices.

TABLE II. Power Consumption Results

Device	**Power consumption (mW)**	
	Mobile Data	***Wi-Fi***
Archos Tablet	723	592
Nexus S	819	718
Nexus 7 (Wi-Fi only)	--	479

C. Optimization Techniques

We propose the following optimization techniques to improve CCT's performance.

- **Avoid multiple GET requests**: This is an extension to CoRE Link specification and another proposed innovation. According to the current implementation of the CoRE Link specification, multiple GET requests are necessary to complete the discovery phase. But we propose that the complete list of resource types and attributes of devices and endpoints can be stored in a database in the M2M gateway. During the discovery phase, the entire list of resource types and attributes can be included in the response from GAE. It will limit network operations and the CPU need to parse the response only once, reducing power consumption.
- **Intelligent filtering at M2M gateway:** The amount of network traffic can be further reduced by examining and filtering out unnecessary endpoint metadata at the M2M gateway.
- **Reducing metadata content**: If the user is interested in a particular set of sensors and their measurements, then the metadata content can be reduced by the gateway. For example, if the user is subscribed to the temperature sensor (in Figure 1), then the SenML metadata is as follows:

```
{"e" : [
        { "n": "Temperature1", "v": "22", "u": "Cel" }],
  "bn" : "http://device1.mydomain.net/",
  "bt" : "1320078429,
  "ver" : 1,
  "ut" : "300"
}
```

  The notations follow the SenML work-in-progress draft. The sensor provides measurements every 5 minutes which are pushed to the mobile client. Instead of pushing the above metadata all the time, some of the constant parts ("bn", "ut", "ver", "u") could be stripped off at the M2M gateway. This will reduce the network data usage and parsing time which will reduce the CPU load and consequently result in prolonging battery life.
- **MQTT based implementation**: Adopting MQTT implementation[4] which is lighter than its HTTP counterpart further reduces the network data consumption and is power efficient. The SenML is currently carried as payload in RESTful implementation. Thus SenML implementation will not be affected at all when carried over MQTT. Just the MQTT implementation has to be developed for the Android application.
- **Brightness Control**: The user interaction time with CCT may vary. But if the interaction time is higher, then the application may reduce the screen brightness during the lifetime of the application [11]. This will also optimize the power consumption.

VII. Use Cases

We present two scenarios which perfectly suits the use of CCT.

- Use of IoT based services in agricultural domain is growing [12, 13]. In this case sensors can be deployed to measure the dryness of the fields continuously. The M2M devices containing such sensors will update the

[3] http://ziyang.eecs.umich.edu/projects/powertutor/

[4]https://www.ibm.com/developerworks/community/blogs/sowhatfordevs/entry/using_mqtt_protocol_advantages_over_http_in_mobile_application_development5?lang=en

farmers through the M2M gateway. If the land becomes too dry, the application will prompt to switch on a motor for irrigation. Controlling the motor corresponds to actuator control part. Such application scenario will be particularly helpful in rural farming lands.

- Hospitals can use the application in connection with a digital thermometer having Bluetooth. Such devices can be fitted on the wrist of the patients and will report their body temperatures to doctors via a gateway. The mobile application will raise an alarm if the temperature rises above a threshold.

The discovery phase, SenML implementations and real time interaction with sensors and actuators are very necessary in the above cases. Thus CCT can be perfect solution as mobile application to the clients. The application can also be used in several other practical scenarios.

VIII. Conclusion

This paper presents an innovative mobile application that enables real time interaction with smart and legacy things through the M2M gateway. The application performs dynamic discovery of the M2M devices and endpoints, benefits from the SenML. The innovations to support actuator control through SenML extensions are discussed and are supported in the application. The Android application registers high CPU load and power consumption which are attributed to intense network operations and parsing the responses. Several optimization techniques are proposed to improve the performance of CCT. The application is also ported for iOS platform and will be tested based on the mentioned two metrics. Currently the architecture only uses HTTP but support for several other protocols like CoAP[5] and DDS[6] will be added to provide wider choice of protocols to the users. We will also extend the discovery phase for M2M gateway discovery which forms the basis of collaboration among such gateway leading to Social IoT. Another important study is to examine the scalability issue to determine the scale of endpoints that can be operated within the architecture. Internet of Things and such mobile applications are vulnerable to new security challenges and end-user privacy [14, 15]. We will definitely experiment with such challenges to provide greater impact with the application.

Acknowledgment

The work is sponsored by French research project WL-Box 4G Pole SCS. The authors would like to thank the project partners from Mios, Gemalto and iQSim for setting up the components of the test scenario.

References

[1] Handong Zhang; Lin Zhu, "Internet of Things: Key technology, architecture and challenging problems," IEEE International Conference on Computer Science and Automation Engineering (CSAE), 2011.

[2] Glitho, R.H., "Application architectures for machine to machine communications: Research agenda vs. state-of-the art," Broadband and Biomedical Communications (IB2Com), 2011 6th International Conference on, pp.1-5, 21-24 Nov. 2011.

[3] Jixuan Zheng; Gao, D.W.; Li Lin, "Smart Meters in Smart Grid: An Overview," Green Technologies Conference, 2013 IEEE, pp.57-64, 4-5 April 2013.

[4] Sanchez, L.; Gutierrez, V.; Galache, J.A.; Sotres, P.; Santana, J.R.; Casanueva, J.; Munoz, L., "SmartSantander: Experimentation and service provision in the smart city," Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on, pp.1-6, 24-27 June 2013.

[5] Liu Tenghong; Yuan Rong; Chang Huating, "Research on the Internet of Things in the Automotive Industry," Management of e-Commerce and e-Government (ICMeCG), 2012 International Conference on, pp.230-233, 20-21 Oct. 2012.

[6] ETSI Technical Specification on Machine-to-Machine Communications; Functional Architecture, ETSI TS 102 690, V2.1.1 (2013-10).

[7] Constrained RESTful Environments (CoRE) Link Format, IETF RFC 6690.

[8] Work-in-Progress Draft on CoRE Interfaces, http://tools.ietf.org/html/draft-shelby-core-interfaces-05

[9] Dalmasso, I.; Datta, S.K.; Bonnet, C.; Nikaein, N., "Survey, comparison and evaluation of cross platform mobile application development tools," Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International Conference on, pp.323-328, 1-5 July 2013.

[10] Datta, S.K.; Bonnet, C.; Nikaein, N.; "Android power management: Current and future trends," Enabling Technologies for Smartphone and Internet of Things (ETSIoT), 2012 First IEEE Workshop on, pp.48-53, 18 June 2012.

[11] Datta, S.K.; Bonnet, C.; Nikaein, N., "Minimizing energy expenditure in smart devices," Information & Communication Technologies (ICT), 2013 IEEE Conference on, pp.712-717, 11-12 April 2013.

[12] Fu Bing, "Research on the agriculture intelligent system based on IOT," Image Analysis and Signal Processing (IASP), 2012 International Conference on, pp.1-4, 9-11 Nov. 2012.

[13] Ji-chun Zhao; Ju-feng Zhang; Yu Feng; Jian-xin Guo, "The study and application of the IOT technology in agriculture," Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol.2, pp. 462-465, 9-11 July 2010.

[14] Lan Li, "Study on security architecture in the Internet of Things," International Conference on Measurement, Information and Control (MIC), 2012.

[15] Hui Suo; Jiafu Wan; Caifeng Zou; Jianqi Liu, "Security in the Internet of Things: A Review," Computer Science and Electronics Engineering (ICCSEE), 2012 International Conference on, vol.3, pp.648-651, 23-25 March 2012.

[5] http://tools.ietf.org/html/draft-ietf-core-coap-18
[6] http://portals.omg.org/dds/