Design of M2M Device Reachability Service Capability

IVAYLO ATANASOV and EVELINA PENCHEVA
Faculty of Telecommunications
Technical University of Sofia
8, St. Kliment Ohridski blvd, 1000 Sofia,
BULGARIA
iia@tu-sofia.bg, enp@tu-sofia.bg

Abstract: - Machine-to-Machine (M2M) communications allow networked devices to exchange information without human intervention. The large variety of devices with diverse characteristics has already resulted in vertical M2M applications based on very heterogeneous forms of platforms, technology, and data models. Interoperability is limited. In order to develop horizontal platforms across different business domains, networks and devices, it is necessary to outline generic capabilities. Service Capabilities provide data mediation functions that may be shared by different applications through application programming interfaces. REpresentational State Transfer (REST) architectural style is adopted for M2M communications. The paper presents an approach to design RESTful Web Services for access to location and presence status information of M2M devices. The Device Reachability Service Capability provides access to device location and allows device presence information to be registered and obtained. Web Service operations are identified by analysis on typical use cases. M2M device reachability information is modelled as REST resources organized in a tree structure. Web Service performance characteristics are evaluated by simulation.

Key-Words: - Machine-to-Machine communications, Service Capability, Web Services, Location, Presence status, Service Level Agreements

1 Introduction

Machine-to-Machine (M2M) communications allow intelligent objects, which are uniquely identified, to capture data from or to control the environment, and to exchange information without human intervention. It is expected that M2M communications will change the businesses like intelligent transport systems, city automation, energy efficiency, and healthcare applications [1], [2]. Forecasts estimate that by the 2018 M2M connected devices will be 19.7% of all connected devices and the mobile M2M communications will represent 6% of all mobile data [3]. This means that the network traffic patterns will be changed dramatically.

One of the main challenges standing in front of M2M technologies is the fragmentation of solutions. Most of the solutions developed and implemented to date address specific application requirements which results in heterogeneous forms of technologies, platforms, and data models [4]. There is a lack of interoperability. Horizontal service platforms have to be used to provide M2M value-added services and to enable reusable, modular and scalable M2M application environment [5], [6].

Service capabilities are software modules that are exposed to M2M applications through the use of application programming interfaces (APIs) [7]. The Web Services (WS) model is based on the assumption that communications are like invocation of remote services whose nature can be ignored [8]. But this model is not suitable for M2M applications, as M2M devices possess constrained computing capabilities. REST (REpresentational State Transfer) is adopted as a method for M2M modelling and implementation. In REST, each physical or logical entity is represented as a resource which has particular state. The resource can be addressed through HTTP Uniform Resource Identifier (URI) and its states can be retrieved and updated. Resources can be created and destroyed respectively.

In this paper, we propose an approach to design RESTfiul Web Services that implement Device Reachability Service Capability and evaluate some performance characteristics concerning WS deployment. The Device Reachability Service Capability provides access to device location and allows device presence information to be registered and obtained.

The paper is organized as follows. Section 2 outlines the related work. Section 3 describes typical use case scenarios, which are used to identify the required WS functionality. Section 4 presents REST resources modelling location and presence status data. Resources are organized in a tree structure and can be uniquely identified. Section 5 describes a model of Service Capability Server (SCS) which provides M2M Device Reachability Service Capability and appears to be a possible bottleneck in the network. The SCS model applies access control for each M2M application provider based on Service Level Agreements (SLAs) in order to prevent congestion. Simulation parameters and results are discussed in Section 6. The conclusion summarizes the contribution.

2 Related Work

The related research concerning access to device location and presence status information addresses specific solutions, but does not study the interoperability-oriented generic functionality. Related studies that deal with device location and presence information include the following.

A recent research, conducted in [9], presents a Connected Devices Interface Generic Enabler providing means to detect and to optimally exploit capabilities and aspects about the status of connected devices, through the implementation of APIs towards device features. The M2M networking-based service coverage framework for post-emergency environments, proposed in [10], performs accurate prediction of user mobility. In [11], it is presented architecture to handle effectively emergency situations which sends the location of incident and contacts the appropriate emergency dealing department automatically. Authors of [12] propose a novel energy-and-memory efficient location update scheme for wireless M2M communications. An on-line summarization mechanism that is able to provide small space representation for massive RFID probabilistic data streams while preserving the meaningfulness of the information is proposed in [13]. High level architectural aspects of access to device information are considered in [14], [15] and [16]. OpenMTC, developed by FOKUS and TU Berlin, is a middleware platform for M2M oriented applications and services which facilitates the development of M2M systems [14]. Open M2M data, presented in [15], is a paradigm in usage and delivery of machine data for sensor networks where the users play a central role. A scalable cloud-based framework for context management able to handle contextual presence information in large distributed environments is described in [16]. A mobile agent network model extended with presence information as a context is presented in [17].

The review on the related implementations shows that they deal either with high level platform or architecture issues or with positioning methods and usage of context data, but do not concern programmability issues. The interoperability issues are considered in terms of frameworks and business models and do not describe the generic primitives/operations and data models in details.

The traffic models and overload control mechanisms developed for human driven communications are not applicable to M2M communications [18]. The last ones feature small and infrequent data transmissions in contrast to web browsing, file transfer and variable bit rate streams. Most researchers in the area focus on overload mechanisms which are radio technology specific. A review on recent advancements in M2M communications in 4G networks and evolution towards 5G is provided in [19]. A method for parameter extraction and channel state prediction in M2M wireless channels is proposed in [20]. In [21], it is presented a dynamic access class barring for M2M communications in LTE networks. A traffic-aware load balancing mechanism for M2M networks using software defined networks is proposed in [22]. Lightweight mobile core network for M2M communications aimed to simplify network attach procedure for M2M devices is described in [23].

While it is important to dimension the network that provides connectivity for M2M devices, there is also a need to evaluate the traffic generated by M2M applications at application level in order to define SLAs for different M2M application providers.

3 Generic Functionality for Access to Device Context Information

In this section, generic Web Service functionality is defined based on the analysis of typical use cases

3.1 Access to location information

The accuracy of location information depends on the application. Additionally, location information may be reported periodically or on demand, and notifications of distance changes between monitored devices may also be available to applications. Periodic location reporting as well as triggered location reporting are provided in case of available subscriptions which require subscription management functionality.

M2M applications run service logic and use available service capabilities. M2M applications may reside in the network (NA), in an M2M gateway (GA) or in an M2M device (DA).

Fig.1 illustrates the message flow when an NA makes a query for M2M device location. Fig.2 illustrates the message flow for subscription to notifications triggered when an M2M device comes in (or goes out of) a specific area, triggered notifications, and subscription termination.

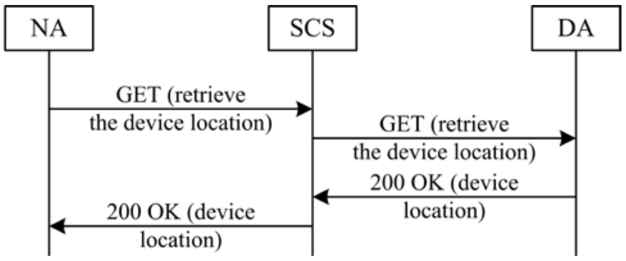
NA
SCS
DA
GET (retrieve the device location)
GET (retrieve the device location)
200 OK (device location)
200 OK (device location)

Fig.1. Query for the device location

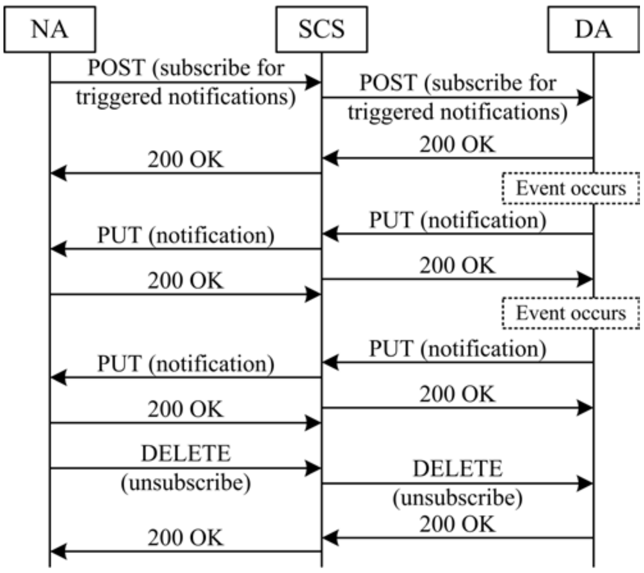

Fig.2. Triggered device location notification

Fig.3 illustrates the message flow for subscription to period notifications about M2M device location, periodic reporting and subscription termination.

An NA may retrieve the location for an M2M device using the *getDeviceLocation* operation. In order to subscribe for triggered location change notifications, the NA uses *startTriggredLocation-Notification*. An DA uses *triggeredLocation-Notification* to notify the NA about a change in the location of the M2M device. Using the *stop-TriggeredLocationNotification* operation the NA may terminate notifications. Similarly, for periodic location reporting *startPeriodicLocationNotification, triggeredLocationNotification* and *stop-TriggeredLocationNotification* operations are used.

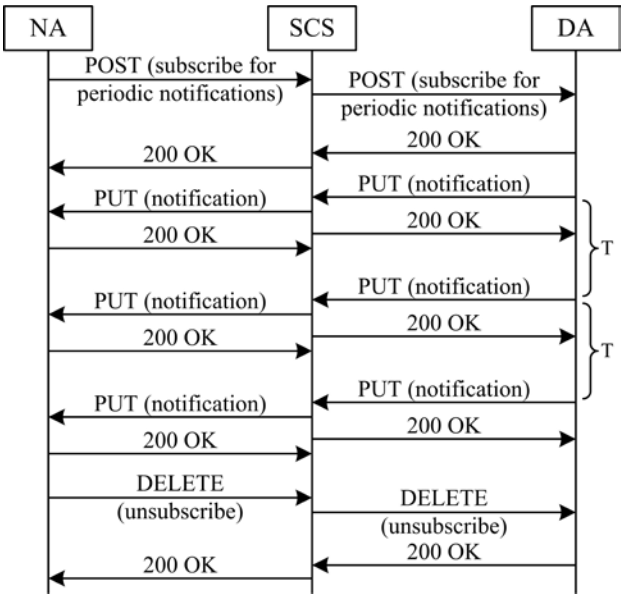

Fig.3. Periodic device location notification

Some applications may be interested in device distance from a location and the *getDeviceDistance* operation may be used for this purpose.

The analysis on the typical use cases shows that the Device Location Web Service should provide access to device location information through:

- request for the location of a device
- notifications of a change in the location of a device
- notification of device location on a periodic basis
- location is expressed through latitude, longitude, altitude and accuracy.

Table 1 lists some of the RESTful WS operations used for access to location information of M2M devices and the respective HTTP methods that can be used for their implementation.

3.2 Access to presence information

In the world of M2M communications, the presence entity is an M2M device which publishes presence information. Presence information consists of a set of attributes that characterize the presence entity such as communication means, environment, current activity, location type etc. A presence attribute contains information about a presence entity. It has a name and a value and can be supplied by any device module including sensors or actuators, application or network module that can be associated to the presence entity.

Table 1. Device Location Web service operations and the corresponding HTTP methods

Device Location Web Service operation	HTTP method
getDeviceLocation	GET
startTriggeredLocationNotification	POST
triggeredLocationNotification	PUT
stopTriggeredLocationNotification	DELETE
startPeriodicLocationNotification	POST
periodicLocationNotification	PUT
stopPeriodicLocationNotification	DELETE
getDeviceDistance	GET

Watcher is an NA which is a presence consumer. The watcher can access presence data by polling or through notifications. Observer is an NA which can retrive information about device watchers.

The *getDevicePresence* operation returns presence data of a presence entity to the watcher. Only presence attributes which the watcher is authorized to access will be returned. The *startPresenceNotification* operation initiates a subscription for notifications of the device presence to the watcher via *statusNotified* operation. The notified presence attributes depend on the watcher subscription status for each presence entity. For each device in a role of presence entity, limited or no presence information may be provided in the resulting notifications which depends on the watcher authorization and allowed access to attributes. The subscription describes the maximum frequency of notifications, period of time notifications occur for, or maximal number of notifications. The *stopPresenceNotification* operation terminates the subscription to presence data notifications. The *statusEnd* operation is delivered to the watcher when the duration or count for notifications has been completed. The *subscriptionEnded* operation is used to notify the watcher that the subscription has terminated, e.g. when an authorized NA decides to block further presence information to that watcher.

The *publish* operation is used by the presence entity to publish data about itself. This data will be filtered and forwarded to the watchers with active subscriptions.

A dedicated observer NA on behalf of the presence entity may periodically call the *detDeviceWatchers* operation to see any watcher(s) subscribing to presence data. In addition, this NA can use *getSubscribedAttributes* operation to retrieve the list of attributes that a specific watcher has subscribed to. In order to manage the notifications for watcher subscriptions, *startDeviceWatcherNotification* and *stopDevice-WatcherNotification* operations may be used. The *notifyDeviceWatchers* operation is called to inform the observer NA of all watcher subscriptions.

An authorized NA may use *updateAutho-rizationRule* operation to change the authorization for certain watchers or the *deleteAuthorizationRule* operation to delete an authorization rule.

Fig.4 shows the sequence diagram of interactions in case of consuming presence information by the watcher.

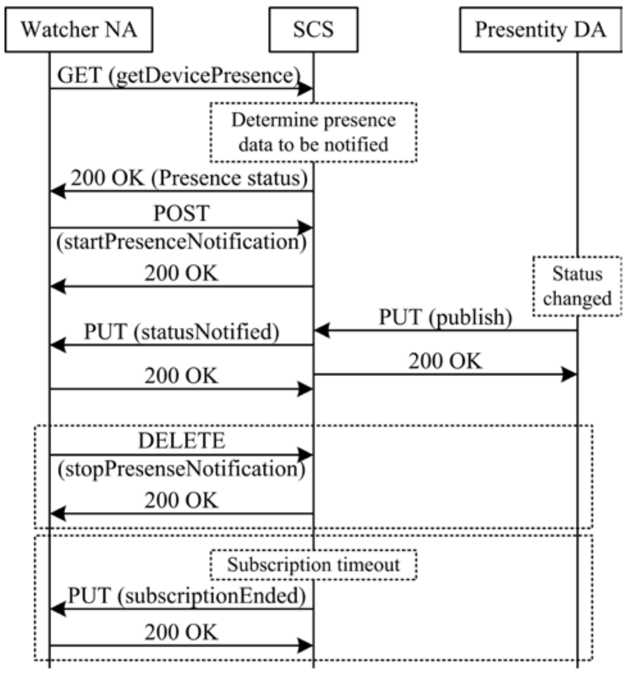

Fig.4. Consumption of presence information

Fig. 5 shows interactions for an observer NA in case of changing an authorization rules and managing notifications about device watchers.

The Device Presence Web Service allows for the presence information to be obtained about an M2M device and to register presence for the device. It is assumed that the typical client of the Web Service is either an M2M DA, or an NA which is observer or consumer of the presence information.

Table 2 lists the Device Presence Web Service operations used for access to presence information of M2M devices and the respective HTTP methods that can be used for their implementation.

4 Device Reachability Resource Structure

All the information related to access to device location and presence status is modeled as resources organized in a tree structure.

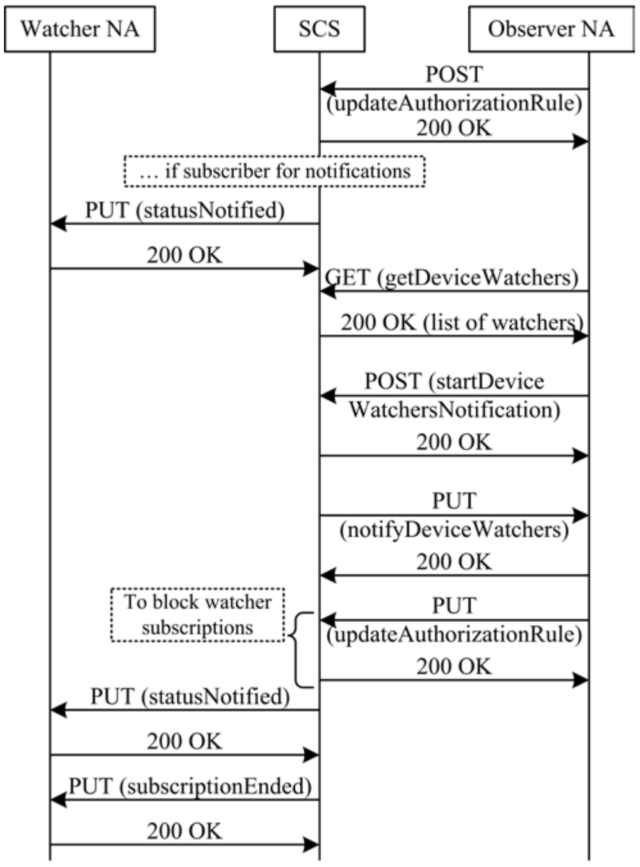
Watcher NA
SCS
Observer NA
POST (updateAuthorizationRule)
200 OK
… if subscriber for notifications
PUT (statusNotified)
200 OK
GET (getDeviceWatchers)
200 OK (list of watchers)
POST (startDeviceWatchersNotification)
200 OK
PUT (notifyDeviceWatchers)
200 OK
To block watcher subscriptions
PUT (updateAuthorizationRule)
200 OK
PUT (statusNotified)
200 OK
PUT (subscriptionEnded)
200 OK

Fig.5. Management of notifications about device watchers

Table 2. Device Presence Web Service operations and the corresponding HTTP methods

Device Presence Web Service operation	HTTP method
getDevicePresence	GET
startPresenceNotification	POST
statusNotified	PUT
stopPresenceNotification	DELETE
statusEnded	PUT
subscriptionEnded	PUT
publish	PUT
getSubscribedAttributes	GET
updateAuthorizationRule	POST
deleteAuthorizationRule	DELETE
startDeviceWatcherNotification	POST
stopDeviceWatcherNotification	DELETE
notifyDeviceWatchers	PUT
getDeviceWatchers	GET

4.1 Model of location information

Location information is presented by latitude, longitude, altitude, accuracy and time stamp. Latitude, longitude and altitude values are expressed as floating point numbers. The accuracy values express the desire of the application for the location information to be provided. The choice of values may influence the price that the service provider would charge. In triggered notifications, a tracking accuracy is defined. Two accuracy values (requested and accepted) may be used. For example, a taxi tracking service that locates the nearest taxi to the client requires fine grained accuracy while coarse grained accuracy may be appropriate for a truck nearing the vicinity of a warehouse. The accuracy of location is provided in meters. In some applications, the maximum age of location information may be useful, e.g. the location information may be cached rather than directly accessed. The maximum acceptance age, in seconds, is expressed in integers.

We followed the ETSI resource structure defined in TS 102690 in modelling location information [7].

Fig.6 shows the resource structure for device location.

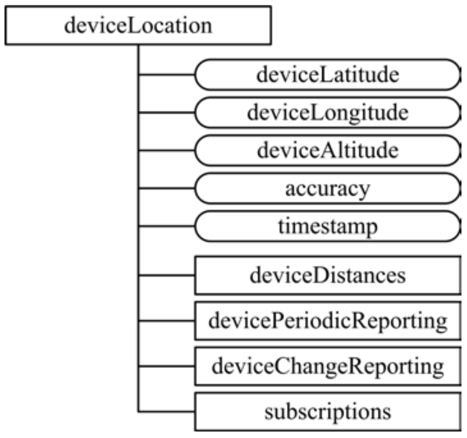

Fig.6 Structure of *deviceLocation* resource

The *deviceLatitude*, *deviceLongitude* and *deviceAltitude* attributes represent the measured device location and the *accuracy* attribute represents the measurement accuracy. The *timestamp* attribute represents the date and time that location was collected.

The <subscriptions> sub-resource of the *device-Location* resource contains a collection of 0..n <*subscription*> resources which represent active subscriptions to location information. Each <*subscription*> resource has *requestedAccuracy* and *acceptedAccuracy* attributes. The *requested-Accuracy* expresses the range in which the subscribed application wants to receive location information. The *acceptedAccuracy* expresses the range that the subscribed application considers to be feasible. If the location cannot be provided within this range, the application prefers not to receive the information. The <*deviceDistances*>, <*devicePeriodicReporting*> and <*deviceChangeReporting*> are sub-resources of the <*deviceLocation*> resource.

Some applications may be interested in device distance from a location. The <*deviceDistances*> collection resource represents the collection of <*deviceDistanceFrom*> resources. The <*deviceDistanceFrom*> resource structure is shown in Fig.7, where the *remoteLatitude* and *remoteLongitude* attributes represent the latitude and longitude of the location to measure from, respectively. The *distance* attribute represents the distance from device to the location specified in meters. The *subscriptions* sub-resource of the <*deviceDistances*> resource contains a collection of 0..n <*subscription*> resources which represent active subscriptions to device distance from the specified location.

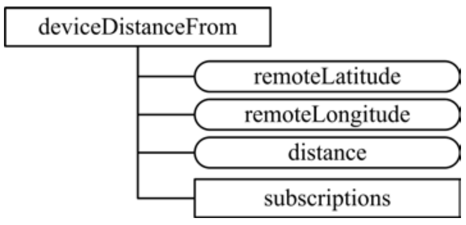

Fig.7 Structure of *deviceDistanceFrom* resource

Notifications of device location may be provided on a periodic basis. The periodic notifications provide location information at an application defined interval. The <*devicePeriodicReproting*> collection resource represents the collection of <*devicePeriodicLocation*> resources. Fig.8 shows the <*devicePeriodicLocation*> resource structure.

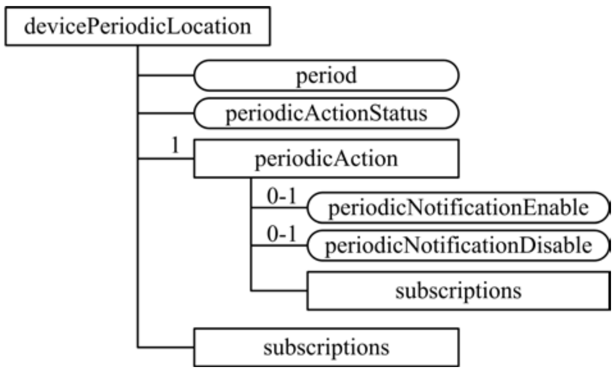

Fig.8 Structure of devicePeriodicLocation resource

The *period* attribute represents the minimum time between notifications (maximum period of notifications can also be considered). The *periodicActionStatus* attribute indicates the status of the action. The <*periodicAction*> resource represents the action. The *periodicNotificationEnable* attribute represents the action that enables the periodic notification. The *periodicNotificationDisable* attribute represents the action that disables the periodic notification.

An application can be notified of a device entering or leaving a geographical area. When a matching event occurs, a notification message will be sent to the application. An application may define a target area and notification criteria e.g. entering the target area or leaving the target area or both. The <*deviceChangeReporting*> collection resource represents the collection of <*deviceLocationChange*> resource. The <*deviceLocationChange*> resource structure for triggered location change notifications is shown in Fig.9.

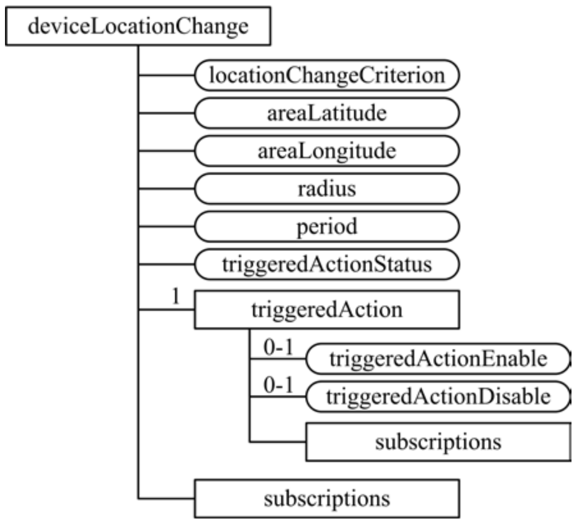

Fig.9 Structure of *deviceLocationChange* resource

The *locationChangeCriteria* attribute is of enumerated type (entering or leaving an area). The *areaLatitude* and *areaLongitude* attributes represent the latitude and longitude of the centre point, and *radius* attribute represents the radius of the circle around the centre point in meters. The *triggeredActionStatus* attribute indicates the status of the action. The <*triggeredAction*> resource represents the action.

4.2 Model of presence status information

The type and structure of presence information depends on the application domain where M2M plays a role e.g. healthcare, transportation, energy, retail, vending, security and surveillance, home/industrial automation and control, etc. M2M device is usually equipped with one or more sensors, and/or one or more actuators. Each device has a communication module with one or more communication channels e.g. for GPRS, WCDMA, LTE, WLAN communications. Some common presence attribute types include the following:

- sensor activity – the device sensor's activity (operational, blocked, damaged, faulty, busy)
- actuator activity - the device actuator's activity
- device place – the device's current place (in operational area, out of operational area, indoor, outdoor).
- communication means – the device's means of communication (UART, RS485, Z-Wave, wM-bus, Zigbee, Bluetooth, 2G/3G/4G cellular, WiFi, Ethernet, PLC).

Fig.10 shows the <*devicePresence*> resource structure (common resource attributes defined in [5] are not shown).

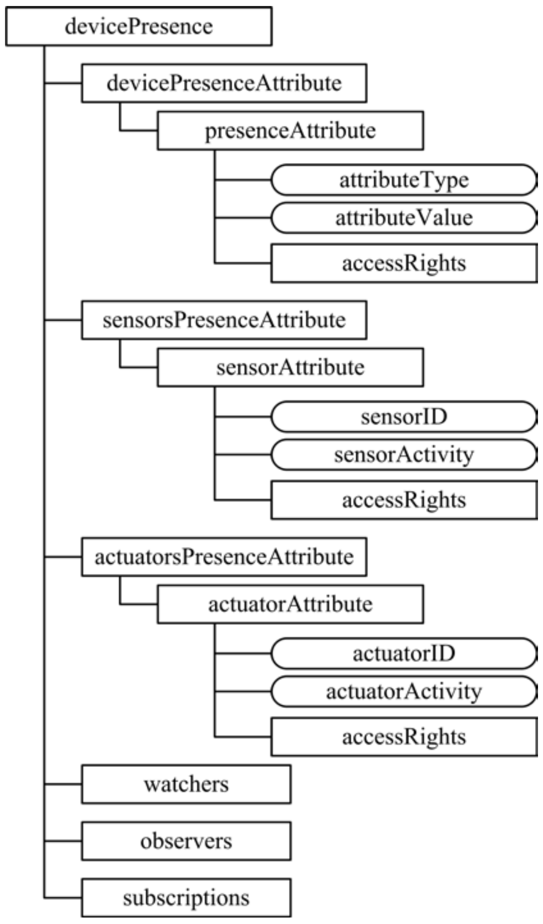

Fig.10 Structure of *devicePresence* resource

The <*devicePresenceAttributes*> resource represents a collection of device presence attributes. Each <*devicePresenceAttribure*> resource has attribute type and value, for example the presence attribute of *communicationMeans* type may have 2G cellular value which means that currently the device uses the operator's 2G network for communications. The <*acceeRights*> resource represents a collection of <*accessRight*> resources, where access rights are defined as "white" list of permissions. The <*sensorPresenceAttributes*> resource and the <*actuatorPresenseAttributes*> resource are collections of sensors' and actuators' presence attributes respectively. The <*watchers*> resource and the <*observers*> resource contain information about watchers and observers respectively.

Fig.11 shows the <*watchers*> resource structure which is a collection of <*watcher*> resources. The *watcherSubscriptionStatus* attribute enumerates the different statuses of the watcher's subscription (authorized, blocked, pending, active, terminated), and the *subscriptionStatusEventTrigger* attribute shows different events that cause a transition in the watcher's subscription status. The <*subscription-Action*> sub-resource is used for management of the watcher's subscription. The structure of the <*observers*> resource is similar to that of the <*watchers*> resource.

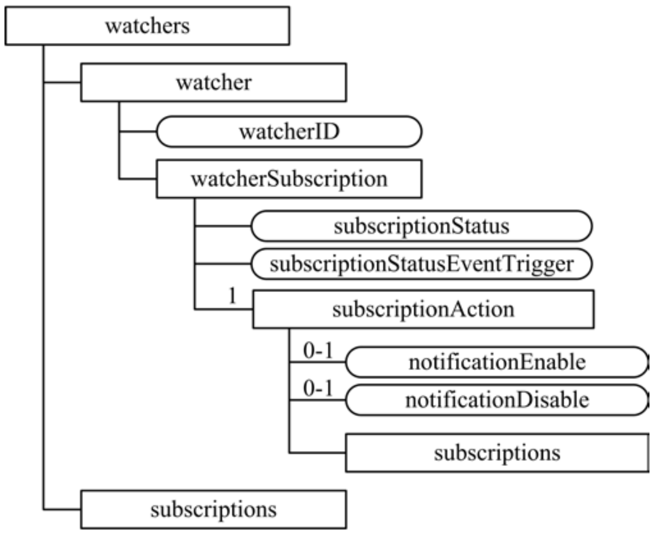

Fig.11 Structure of watchers resource

5 Service Capability Server Model

Fig.12 shows a simplified architecture for M2M communications. The Service Capability Server is an entity controlled by the network operator or by an M2M Service Capability Provider. The SCS offers capabilities for use by one or multiple M2M applications.

An M2M device can host one or multiple DAs. The corresponding M2M NAs in the external network are hosted on one or multiple Application Servers, which may be provided by third party.

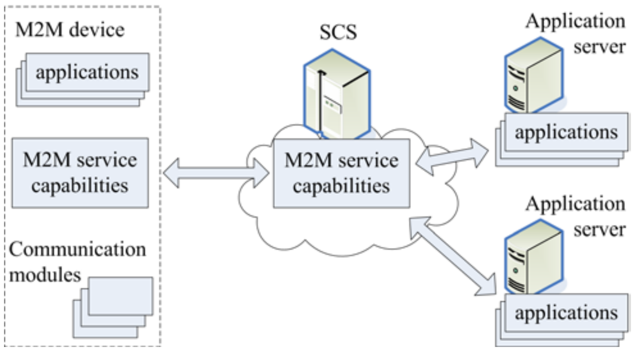

Fig.12 Simplified M2M functional architecture with focus on the third party access

The contract (SLA) between an Application Provider (AP) and the M2M Service Capability Provider (SP) defines constraints that have to be fulfilled. The constraints include the peak and average number of network application requests and M2M device notifications that should be accepted per time unit. Access control is applied to each M2M Application Provider in order to prevent the SCS from overloading.

The components used for access rate restriction are of two types, namely Token Bucket mechanism and message filtering. The well known Token Bucket mechanism uses tokens to grant access where tokens arrive at constant rate and are distributed among arriving requests. The message filtering is used to recognize a request in order to apply different access control for POST, GET, PUT and DELETE requests as specified in SLA. The SCS state is represented by the number of requests that the accepted request queue holds until being forwarded.

Fig.13 shows the SCS access control model built from message filtering (F_i) and token bucket (B_i) components.

Each bucket $B_i(T, \rho, \mu)$ is described by the upper limit of tokens T, the token arrival rate ρ_i and the current number of tokens μ_i. The simplest possible model for message filter is the one that checks whether given message belongs to a class. Assuming that for given interval (t_{k-1}, t_k] the flow of messages at the filter input is $N(t_k)$ and the class to filter is set to be c then

$$F_c(t_k) = \sum_{\forall m \in N(t_k)}^{n} I(m \in c) \qquad (1)$$

where $I(s)$ is 1 when s is true, and 0 otherwise.

It is trivial that the part of the flow filtered out is

$$\hat{F}_c(t_k) = N(t_k) - F_c(t_k) \qquad (2)$$

The queue model (not shown in Fig.13() is described with respect to the possible loss introduced by its finite length Q. So, it is normal to assume that the initial queue state is empty i.e. $q(t_0) = 0$, and in case it is full, then the queue state becomes $q(t_k) = Q$.

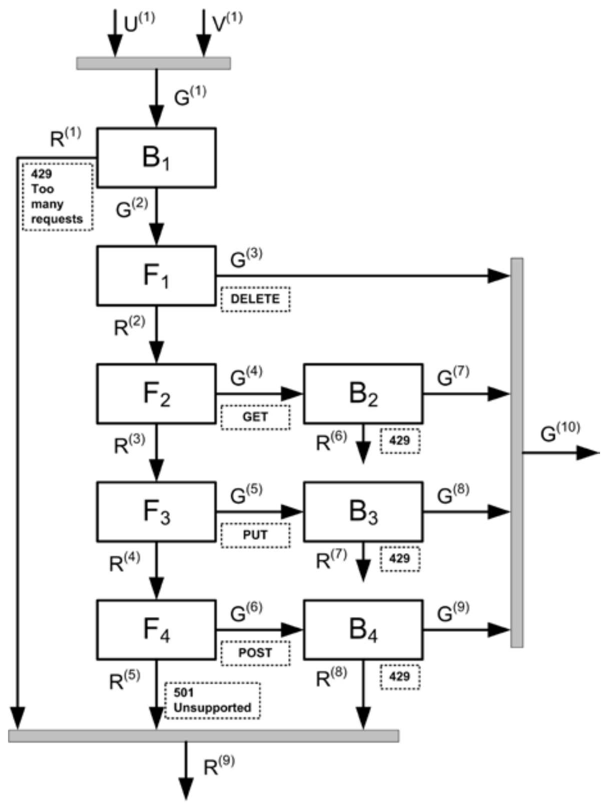

Fig.13 A model of SCS's access controller

Then the loss at the queue for given interval (t_{k-1}, t_k] is formed by all the messages from the arriving flow that come when queue happens to be full, and this might be expressed by

$$L(t_k) = \sum_{t_{k-1} < \tau_i \le t_k} I(m\ (\tau_i) \in A(t_k)).I(q(\tau_i) \equiv Q) \qquad (3)$$

The incoming request flow is generated by M2M network applications (U) hosted by a service provider and Device/ Gateway M2M applications (V). The granted and the rejected requests are denoted by G and R respectively. The first token bucket B_1 limits the total number of requests. The request is rejected due to violation of SLA, an HTTP 429 *Too many requests* response is sent. The filter F_1 passes DELETE requests all of which are admitted as they are used to release resources. The filter F_2 passes GET requests which are then limited by the token bucket B_2. The filter F_3 passes PUT requests which are then limited by the token bucket B_3. Finally, the filter F_4 passes POST requests which are then limited by the token bucket B_4.

The request flow is observed in regular intervals (t_{k-1}, t_k]. The flows are expressed by the following equations:

$$G^{(1)}(t_k) = U^{(1)}(t_k) + V^{(1)}(t_k) \tag{4}$$

$$G^{(2)}(t_k) = \min(\mu_1(t_{k-1}) + \min(T_1 - \mu_1(t_{k-1}), \rho_1 \Delta t_k), G^{(1)}(t_k)) \tag{5}$$

$$R^{(1)}(t_k) = G^{(1)}(t_k) - G^{(2)}(t_k) \tag{6}$$

$$G^{(3)}(t_k) = \sum_{\forall m_i \in G^{(2)}(t_k)} I(m_i \in c_1) \tag{7}$$

$$R^{(2)}(t_k) = G^{(2)}(t_k) - G^{(3)}(t_k) \tag{8}$$

$$G^{(4)}(t_k) = \sum_{\forall m_i \in R^{(2)}(t_k)} I(m_i \in c_2) \tag{9}$$

$$R^{(3)}(t_k) = R^{(2)}(t_k) - G^{(4)}(t_k) \tag{10}$$

$$G^{(5)}(t_k) = \sum_{\forall m_i \in R^{(3)}(t_k)} I(m_i \in c_3) \tag{11}$$

$$R^{(4)}(t_k) = R^{(3)}(t_k) - G^{(5)}(t_k) \tag{12}$$

$$G^{(6)}(t_k) = \sum_{\forall m_i \in R^{(4)}(t_k)} I(m_i \in c_4) \tag{13}$$

$$R^{(5)}(t_k) = R^{(4)}(t_k) - G^{(6)}(t_k) \tag{14}$$

$$G^{(7)}(t_k) = \min(\mu_2(t_{k-1}) + \min(T_2 - \mu_2(t_{k-1}), \rho_2 \Delta t_k), G^{(4)}(t_k)) \tag{15}$$

$$R^{(6)}(t_k) = G^{(4)}(t_k) - G^{(7)}(t_k) \tag{16}$$

$$G^{(8)}(t_k) = \min(\mu_3(t_{k-1}) + \min(T_3 - \mu_3(t_{k-1}), \rho_3 \Delta t_k), G^{(5)}(t_k)) \tag{17}$$

$$R^{(7)}(t_k) = G^{(5)}(t_k) - G^{(8)}(t_k) \tag{18}$$

$$G^{(9)}(t_k) = \min(\mu_4(t_{k-1}) + \min(T_4 - \mu_4(t_{k-1}), \rho_4 \Delta t_k), G^{(6)}(t_k)) \tag{19}$$

$$R^{(8)}(t_k) = G^{(6)}(t_k) - G^{(9)}(t_k) \tag{20}$$

$$G^{(10)}(t_k) = G^{(3)}(t_k) + G^{(7)}(t_k) + G^{(8)}(t_k) + G^{(9)}(t_k) \tag{21}$$

$$R^{(9)}(t_k) = R^{(1)}(t_k) + R^{(5)}(t_k) + R^{(8)}(t_k) \tag{22}$$

$$R^{(10)}(t_k) = \sum_{t_{k-1} < \tau_i \le t_k} I(m(\tau_i) \in G^{(10)}(t_k)).I(q(\tau_i) \equiv Q) \tag{23}$$

The static approach of rate-conditioning is stable but restrictive and this becomes obvious especially in case of event-driven notification scheme when considerable amount of notifications might get discarded while the primary bucket contains tokens.

A partial relaxation of the restriction might be introduced by intra-SLA short-term rate redistribution between secondary buckets. Assuming that the arrival process is at least short-term stationary and A denotes the number of events, the simplest form of rate redistribution is

$$\rho^{(x)}(t_{k+1}) = \rho_g^{(x)} + \Delta\rho \tag{24}$$

$$\rho^{(y)}(t_{k+1}) = \rho_g^{(y)} - \Delta\rho \tag{25}$$

where ρ_g is static guaranteed rate per request type.

The index x is the second level bucket index subject to

$$x = \arg\max_j R_j \tag{26}$$

and y is the bucket index such that

$$S_y = \mu^{(y)}(t_k) + \rho_g^{(y)} \Delta t_k - T^{(y)} - A^{(y)} \tag{27}$$

is a positive maximum.

If such positive maximum exists, with no change of total tokens amount for the temporarily amended SLA, the donor's contribution becomes

$$\Delta\rho = \rho_g^{(y)} - \frac{T^{(y)} + A^{(y)} - \mu^{(y)}(t_k)}{\Delta t_k} \tag{28}$$

and it is re-evaluated before each time unit.

The access control is applied for each service provider j. Let us denote by C the SCS's capacity and by N the number of service providers.

Then the SCS utilization might be formulated by

$$\eta = \frac{1}{C.(t_k - t_0)} \sum_{j=1}^{N} \sum_{i=1}^{K} (G_j^{(1)}(t_i) - (R_j^{(9)}(t_i) + R_j^{(10)}(t_i))) \tag{29}$$

where K defines the integral time period.

6 Simulation Results

The simulation is done on simplified M2M SCS model with three classes of requests. The parameterization of the simulation model is provided by a mobile operator. The capacity of the M2M SCS is 800 requests per second. The behaviour of each M2M service provider is modelled by Markov Modulated Poisson Process [24]. New application requests are generated according to four-state MMPP. Changes between different states are uniformly distributed and occur according to Poisson Process with mean 4*s*. The time intervals between POST requests (subscriptions to location notifications), between PUT requests (location notifications) and between GET requests (location retrievals) are exponentially distributed as the arrival process in the WS context with mean 200 *s*, 50 *s*, 50 *s* respectively. The token rate is equal to the guaranteed rate and the bucket size is determined by the peak rate. Initially, $\mu_i(t_0) = T_0$. The length of interval for observation (t_{k-1}, t_k] is set to 100 *ms*. The mean processing time for a single request/ response is 5 *ms*.

The aim of simulation is to evaluate the M2M SCS utilization setting different values of guaranteed rates and fixed peak rates. The guaranteed rates for a given M2M service provider define the constraints: GR_1 for preventing the M2M SCS from overloading, GR_2 for the rate of POST requests, GR_3 for the rate for PUT requests, and GR_4 for the rate of GET. The processing capacity of the M2M SCS must be distributed between different types of messages, where the overall message peak rate (PR_1) must be spread between POST requests (PR_2), PUT requests (PR_3), GET requests (PR_4), and DELETE requests.

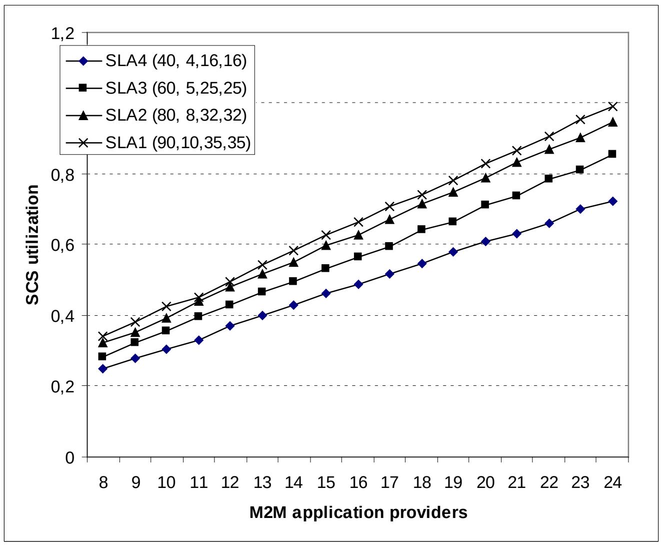

Fig.14 Utilization of M2M SCS versus the number of M2M application providers

The simulation is run in a space of SLAs. The SLA$_j$ for j-th M2M application provider consists of the tuple $(GR_1^j, GR_2^j, GR_3^j, GR_4^j)$ for every token bucket of j-th access controller. The strictest restrictions are posed on the POST requests as they generate new subscriptions to notifications. The values of peak rates PR_1 =100, PR_2 =12, PR_3 =38, PR_4 =38 are limited by the capacity of M2M SCS.

Fig.14 summarizes the outcome of the simulation. The M2M SCS utilization is evaluated as a function of the number of M2M application providers and guaranteed rates.

The simulation results show that the M2M SCS utilization depends both on the number of M2M application providers and on the specific values of rates in SLAs. In case the congestion threshold value is set to 80%, it is most likely that the appropriate choice is to have 22 M2M application providers applying second type or third type of SLA.

7 Conclusion

Horizontal service platforms accommodate various M2M applications providing reusable functional bricks through application programming interfaces. The paper studies generic functionality required to support M2M service capabilities that provide access to location and presence information of M2M devices. An approach to design RESTful Web Services is presented.

Different use case scenarios related to usage of contextual information are considered in order to identify basic RESTful Web Service operations. The operations allow delivery of location information on demand, periodically and upon event and obtaining presence status information. These operations are mapped onto HTTP requests. The semantic information related to device reachability is modelled and the information model explicitly determines the structure of data exchange between M2M applications. All data related to device location and presence status is represented as a set of resources. The resources are organized in a tree structure and may be manipulated through their values. As each of the resources is uniquely addressable, it can be accessed using standard HTTP methods or CoAP (Constrained Application Protocol) primitives. The logic behind the usage of the Web Services is that the same data structures may be used to access and store data, and different applications may exchange data in interoperable fashion.

Moreover, context management to a group of M2M devices is not currently considered. In a future work, we plan to extend the WS functionality with access to location and presence status information of a group of devices.

Deployment aspects of the designed RESTful WS are considered. By evaluation of the load of the Service Capability Server providing access to device location and presence status information, the values of quality of service related parameters in SLA contracts between M2M application providers and the provider of M2M service capabilities are determined.

The proposed approach enables separation of M2M service capabilities and application logic. It simplifies configuration of M2M applications which may be designed once and deployed anywhere.

References:

[1] N. Shakhakarmi, “Next Generation Wearable Devices: Smart Health Monitoring Device and Smart Sousveillance Hat using Device to Device (D2D) Communications in LTE Assisted Networks”, *WSEAS Transactions on Communications*, vol.14, 2015, pp.241-255.

[2] M. Collotta, A. Messineo, G. Nicolosi, G. Pau, “A Self-Powered Bluetooth Network for Intelligent Traffic Light Junction Management”, *WSEAS Transactions on Information Science and Applications,* vol.11, 2014, pp.12-23.

[3] C. Pereora, A. Aguiar, “Towards Efficient Mobile M2M Communications: Survey and Open Challenges”, *Sensors*, vol. 14, 2014, pp.19582-19608.

[4] J. Kim, J. Lee, J. Kim, J. Yun, "M2M Service Platforms: Survey, Issues, and Enabling Technologies", *IEEE Communications Surveys & Tutorials*, vol.16, no.1, 2014, pp. 61-76.

[5] J. Latvakoski, M.B. Alaya, H. Ganem, B. Jubeh, A. Iivari, J. Leguay, J. M. Bosch, N. Granqvist, "Towards Horizontal Architecture for Autonomic M2M Service Networks", *Future Internet*, vol.6, 2014, pp.261-301.

[6] J. Latvakovski, A. Iivari, P. Vitic, B. Juben. M. Alaya, T. Monteil, Y. Lopez, G. Talavera, J. Gonzalez, N. Granqvist, M. Kellil, H. Ganem, T. Vaisanen, "A survey on M2M Service Networks", *Computers*, vol.2, 2014, pp.130-173.

[7] ETSI TS 102 690, "Machine-to-Machine communications (M2M); Functional architecture", 2011.

[8] M. D. Lakshmi, J. P. M. Dhas, "A Hybrid Approach for Discovery of OWL-S Services Based on Functional and Non-Functional Properties", *WSEAS Transactions on Communications*, vol.14, 2015, pp.62-71.

[9] P. Garino, L. Zuccaro, G. Oddi, A. Paolo, A. Simeoni, "Future Internet: the Connected Device Interface Generic Enabler", *WSEAS Transactions on Communications*, vol.14, 2015, pp.226-234.

[10] N. A. Surobhi, A. Jamalipour, "M2M-Based Service Coverage for Mobile Users in post-Emergency Environments," *IEEE Transactions On Vehicular Technology*, vol. 63, no. 7, 2014, pp. 3294-3303.

[11] M. Aazam, P. P. Hung, E. N. Huh, "M2M Emergency Help Alert Mobile Cloud Architecture", *IEEE 29th International Conference on Advanced Information Networking and Applications Workshops*, Conference proceedings, 2015, pp. 500-505.

[12] R.H. Gau, C. P. Cheng, "Optimal Tree Pruning for Location Update in Machine-to-Machine Communications", *IEEE Transactions on Wireless Communications*, vol.12, issue 6, 2013, pp. 2620-2632.

[13] R. Haider, F. Mandreoli, R. Martoglia, "Effective Aggregation and Querying of Probabilistic RFID Data in a Location Tracking Context", *WSEAS Transactions on Information Science and Applications*, vol.12, 2015, pp.148-160.

[14] S. Wahle, T. Magedanz, F. Schulze, "Demonstration of OpenMTC – M2M Solutions for Smart Cities and the Internet of Things", Available at: http://Www.Ieeelcn.Org/Prior/LCN37/Lcn37demos/Lcndemos12_Wahle.Pdf, 2013.

[15] H. Coskun, T. Pfeifer, A. Elmangosh, A. A. Hezmi, „Open M2M Data – Position Paper", *IEEE International Workshop on Machine to Machine Communications Interfaces and Platforms*, 2103, pp. 904-911.

[16] B. Chihani, E. Bertin, N. Crespi, "Enhancing M2M communication with cloud-based context management", *IEEE International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST)*, Conference proceedings, 2012, pp.36-41.

[17] M. Kusek, I. Lovrek, H. Maracic, "Rich Presence Information in Agent Based machine-to-Machine Communications", *Procedia Computer Science, 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems (KES'2013)*, Elsevier, 2013, pp.321-329.

[18] Y. C. Chang, "Study of Overload Control problem for Intelligent LTE M2M Communication System", *Advances in Smart Systems Research*, vol.3, no.3, 2013, pp.44-48.

[19] R. Ratasuk, A. Prasad, Z. Li, A. Ghosh, M. Uusitalo, "Recent Advancements in M2M Communications in 4G Networks and Evolution Towards 5G", *International Conference on Intelligence in Next generation Networks, ICIN'2015*, Conference proceedings, 2015, pp.1.-6.

[20] R. Adeogun, "A Method for Parameter Extraction and Channel State Prediction in Mobile-to-Mobile Wireless Channels," *WSEAS Transactions on Communications*, vol.13, 2014, pp.665-671.

[21] S. Duan, V. S. Mansouri, V. Wong, "Dynamic Access Class Barring for M2M Communications in LTE Networks", *Wireless Networking Symposium, Globecom'2013*, Conference proceedings, 2013, pp.4747-4753.

[22] Y. J. Chen, Y.H. Shen, L.C. Wang, "Traffic aware Load Balancing for M2M Networks Using SDN", *IEEE International Conference on Cloud Computing Technology and Science*, Conference proceedings, 2014, pp.668-671.

[23] T. Taleb, A. Ksentini, A. Kobbane, "Lightweight Mobile Core Networks for machine Type Communications", *IEEE Access*, 2014, vol.2 pp.1128-1137.

[24] Zhang Qi-zhi "On overload control of parlay application server in next generation network", *Journal of China Universities of Posts and Telecommunications*, vol. 15, issue 1, 2008, pp.43-47.