DE GRUYTER OPEN

Print ISSN: 1312-2622; Online ISSN: 2367-5357
DOI: 10.1515/itc-2016-0019

Model Aspects of Quality of Service Management in Machine to Machine Communications

I. Atanasov, A. Nikolov, E. Pencheva

Key Words: *Device management; light-weight M2M; bisimulation; autonomous agent.*

Abstract. *Machine-to-Machine communications (M2M) allow connected devices to exchange data and perform actions without human intervention. Different M2M applications have different quality of service (QoS) requirements. The dynamic QoS control is critical for applications like video surveillance, transportation services, and industrial control. The provided QoS depends on device connectivity. Connectivity management provides centralized management of M2M devices and connections. The paper studies model aspects of QoS management for connected M2M devices by means of connectivity management. Models for connectivity management are suggested, formally described and verified. The capabilities for device connectivity management are used to model the behavior of autonomous agent for dynamic QoS control.*

1. Introduction

Machine type communications (M2M) is pervasive technology which is applied in segments such as home automation, healthcare, security and safety, smart energy, education, automotive etc. Different M2M applications have different requirements to quality of service (QoS) [1,2]. Some QoS requirements depend on the transported media e.g. video surveillance, while other QoS requirements are more specific and are not typical for data services between humans, e.g. transfer delays for industrial control. The availability of high-speed, low-latency data communications provided by four generation (4G) technology creates opportunities to overcome some challenges [3,4]. In [5], 3GPP has specified features for machine-type communications in 4G networks that include functionalities for reducing power consumption, or for avoiding network congestion, e.g. due to a large number of devices, all trying to communicate at once. Much of intensive research work related to QoS in M2M communication is mainly focused on constraints of radio access networks, while the efforts regarding to application control on QoS generally consider business aspects or high level architectural frameworks [6,7].

The provided QoS may depend on the bearer used by the device to connect to the network. Devices may be connected using cellular bearers such 2G, 3G and 4G technologies, wireless bearers like Wireless Local Area Networks (WLAN) or Bluetooth, or may use wireline ones as Ethernet, Digital Subscriber Lines or Power Line Connections. Hence, in case of several available bearers, connectivity management may be regarded as one of the means to provide the required QoS.

Lightweight M2M (LWM2M) is a protocol from the Open Mobile Alliance (OMA) for M2M device management [8]. It defines device management procedures between a LWM2M server and a LWM2M client, which is located in a device. The protocol is used to create device management solutions that apply the approach of software defined networks [9,10]. The proposed solutions based on LWM2M consider high level architectural aspects and do not provide details on behavioral models that follow the M2M device management procedures.

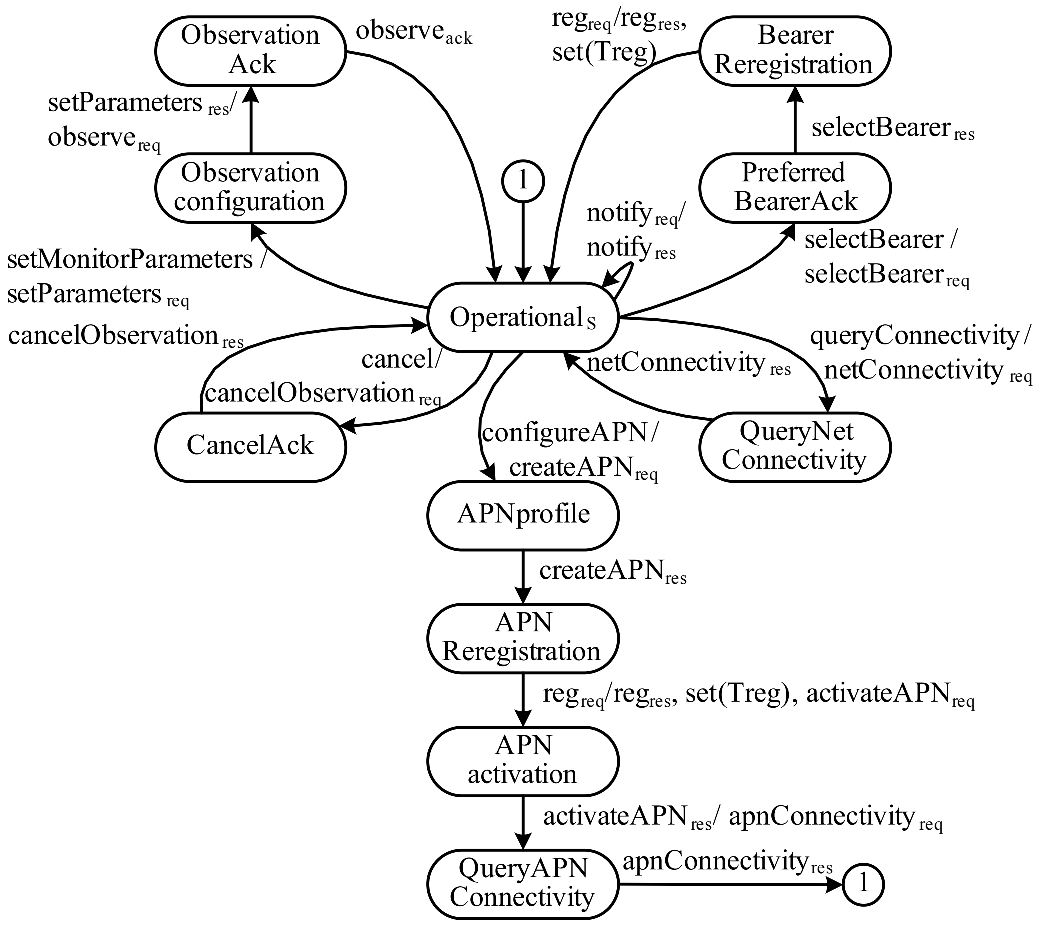
Our approach to QoS management in M2M communications is based on the capabilities for connectivity management provideÄ by LWM2M. We use OMA traps defined for device management which allow diagnostics and monitoring activities [11]. As far as LWM2M specifications provide a high level framework for remote device management [12], we suggest models associated with client-side and server-side behavior necessary to utilize the event monitoring capabilities on the mobile devices. The usage of information captured by diagnostics and monitoring activities is modeled by autonomous agent that controls the QoS provided on device connections.

The paper is organized as follows. In Section 2, available test cases for LWM2M and specific diagnostics and monitoring trap events are described which forms the foundation of our models. In Section 3 and Section 4 client and server behavioral models of the connectivity management are presented and formally described. In Section 5, we propose a method for formal verification of the models based on the concept of weak bisimulation. In Section 6, a model of autonomous agent for QoS manage-ment is provided and formally described using temporal logic. Section 7 provides a discussion on the benefits of the proposed approach to QoS management in M2M communications. The conclusion summarizes the contribution and outlines the directions for future work.

2. OMA Connectivity Management

The OMA LWM2M protocol is targeted at constrained devices with low power microcontrollers and limited amount of memory, as well as at more powerful embedded devices. It is a protocol between a server located in a public or private data center, and a client which resides on the device. The interface between the LWM2M client and server allows efficient device management.

The server may observe line voltage or signal strength



Observation Ack
observe_ack
reg_req/reg_res, set(Treg)
Bearer Reregistration
setParameters res/ observe req
selectBearer res
Observation configuration
1
Preferred BearerAck
notify req/ notify res
setMonitorParameters / setParameters req
selectBearer / selectBearer req
Operational S
cancelObservation res
cancel/ cancelObservation req
queryConnectivity / netConnectivity req
netConnectivity res
CancelAck
configureAPN/ createAPN req
QueryNet Connectivity
APNprofile
createAPN res
APN Reregistration
reg req/reg res, set(Treg), activateAPN req
APN activation
activateAPN res/ apnConnectivity req
QueryAPN Connectivity
apnConnectivity res
1

Figure 1. Connectivity observation model as seen by the server

at the device side. For this purpose, the server establishes an observation relationship with the device in order to set the observation policy. The device sends periodical or triggered reports with requested information until the server cancels the observational relationship. The server may query about multiple network parameters related to connectivity on the device, including used bearer and available bearers. If for example the device has cellular network connectivity and supports WLAN connectivity, and WLAN coverage is available, then the server may control on bearers by requesting bearer selection with preferred WLAN communication bearer. The server may also create and enable a new Access Point Name (APN) profile. Typical sequence of procedures performed by the server and device in the context of connectivity management is as follows.

1. The server establishes an observation relationship with the device to acquire periodical or triggered notifications about QoS parameters.
2. The device sends periodical or triggered notifications about requested QoS parameter values.
3. The server queries about used and available network bearers.
4. The server initiates bearer selection eventually.
5. The server queries about connectivity parameters.
6. The server creates and enables a new APN profile.
7. The server cancels observation.

In the next sections, we model the behavior at the client-side (device) and server-side associated with the above listed procedures.

OMA DiagMon Trap Events specification defines a number of standardized traps [14]. Geographic trap goes to active when a device enters into a specific geographic area. Whenever the device leaves that specific geographic area, the trap goes to inactive. Collection of measurement data (e.g. signal strength and quality) is important for the connectivity management when the device communi-cates over the wireless network. Received power trap can help the connectivity optimization process by triggering the server when the received power of the device drops below the server-specific value. Whenever a device's received power drops below a server-specified value (TrapActivePower), it causes this trap to go active. Alternatively, when device senses power rises above another server-specified value (TrapInactivePower), it causes this trap to go inactive. In cases that the trap goes active or inactive, the device notifies the registered server. The device can have several instances of this kind of trap to monitor various network types.

OMA defines also Call drop trap, Data speed trap, and Quality of service (QoS) traps. Whenever a call drop occurs in the predefined period, it causes Call drop trap to go active. Whenever an uplink or downlink average data speed reaches the lower or higher threshold value, it causes Data speed trap to react. If the device exposes QoS metrics functionality then the QoS trap is enabled in case a measured parameter value is equal or greater than a predefined QoSLowerThreshold value or it is minor or equal than a predefined QoSUpperThreshold value. All these traps may trigger bearer change.

OMA defined diagnostics and monitorring trap events are used in our connectivity models to configure observation parameters and for triggering the activity of an autonomous agent for QoS management presented in Section 6.

3. Server's View on Connectivity Management

The connectivity management model as seen by the server is shown in *figure 1.*

In Operational$_S$ state, the device is registered and operational. In ObservationConfiguration state, the server sets obser-vation policy in the device. In ObservationAck state, the server waits for acknowledgment that the observation is active. In QueryNetConnectivity state, the server has sent a query for device connectivity parameters and waits for the requested information. In PreferredBearerAck state, the server has requested preferred bearer selection and waits for the acknowledgement. In BearerReregistration state, the server waits for device re-registration after bearer selection. In APNprofile state, the server has requested creation of new APN profile and waits for acknowledgement. In APNReregistration state, the server waits for device re-registration after new APN profile selection. In APNactivation state, the server has activated the new APN and waits for acknowledgement. In QueryAPNConnectivity state, the server waits for the requested information about APN connectivity. In CancelAck state, the server has canceled the observational relationship and waits for acknowledgement.

We use the notation of Labeled Transition System (LTS) to formally describe the model.

By $CM_S = (S_S, Act_S, \rightarrow_S, s_0^S)$ it is denoted an LTS representing the server's view on connectivity management state model as follows:

S_S = {Operational$_S$, ObservationConfiguration, ObservationAck, QueryNetConnectivity, PreferredBearerAck, BearerReregistration, APNReregistration, APNprofile, APNactivation, QueryAPNConnectivity, CancelAck };

Act_S= {queryConnectivity, netConnectivity$_{res}$, setMonitorParameters, notify$_{req}$, setParameters$_{res}$, observe$_{ack}$, selectBearer, selectBearer$_{res}$, reg$_{req}$, configureAPN, createAPN$_{res}$, activateAPN$_{res}$, apnConnectivity$_{res}$, cancel, cancelObservation$_{res}$};

$\rightarrow_S = \{ \tau_1^S, \tau_2^S, \tau_3^S, \tau_4^S, \tau_5^S, \tau_6^S, \tau_7^S, \tau_8^S, \tau_9^S, \tau_{10}^S, \tau_{11}^S, \tau_{12}^S, \tau_{13}^S, \tau_{14}^S, \tau_{15}^S, \tau_{16}^S \}$;

s_0^S = { Operational$_S$ }.

where

τ_1^S = (Operational$_S$ setMonitorParameters ObservationConfiguration),

τ_2^S = (ObservationConfiguration setParameters$_{res}$ ObserveAck),

τ_3^S = (ObserveAck observe$_{ack}$ Operational$_S$),

τ_4^S = (Operational$_S$ notify$_{req}$ Operational$_S$),

τ_5^S = (Operational$_S$ queryConnectivity QueryNetConnectivity),

τ_6^S = (QueryNetConnectivity netConnectivity$_{res}$ Operational$_S$),

τ_7^S = (Operational$_S$ selectBearer PreferredBearerAck),

τ_8^S = (PreferredBearerAck selectBearer$_{res}$ BearerReregistration),

τ_9^S = (BearerReregistration reg$_{req}$ Operational$_S$),

τ_{10}^S = (Operational$_S$ configureAPN APNprofile),

τ_{11}^S = (APNprofile createAPN$_{res}$ APNReregistration),

τ_{12}^S = (APNReregistration reg$_{req}$ APNactivation),

τ_{13}^S = (APNactivation activateAPN$_{res}$ QueryAPNConnectivity),

τ_{14}^S = (APNactivation apnConnectivity$_{res}$ Operational$_S$),

τ_{15}^S = (Operational$_S$ cancel CancelAck),

τ_{16}^S = (CancelAck cancelObservation$_{res}$ Operational$_S$).

Next section presents the device view on connectivity management.

4. Device's View on Connectivity Management

The connectivity management model as seen by the device is shown in *figure 2.*

In Operational$_D$ state, the device is registered and operational. In this state, the server may set the observation policy and activate observation, as well as it may cancel observation. In Operational$_D$ state, the device sends responses of queries on network connectivity. In NotifyAck state, the device has notified the server about requested information and waits for response. In UpdateBearer state, the device is in a process of used network bearer switching and re-registration. In APNconfiguration state, the device is in a process of creation and enablement of a new APN profile.

By $CM_D = (S_D, Act_D, \rightarrow_D, s_0{}^D)$ it is denoted an LTS representing the device's view on connectivity management state model as follows:

S_D = { Operational$_D$, NotifyAck, UpdateBearer, APNconfiguration };
Act_D= { setParamaters$_{req}$, observe$_{req}$, netConnectivity$_{req}$, cancelObservation$_{req}$, T$_{mon}$, trigger, notify$_{res}$, selectBearer$_{req}$, reg$_{res}$, activateAPN$_{req}$, create$_{req}$, apnConnectivity$_{res}$ };
$\rightarrow_D = \{ \tau_1^D, \tau_2^D, \tau_3^D, \tau_4^D, \tau_5^D, \tau_6^D, \tau_7^D, \tau_8^D, \tau_9^D, \tau_{10}^D, \tau_{11}^D, \tau_{12}^D, \tau_{13}^D \}$
$s_0{}^D$ = { Operational$_D$ },
where

τ_1^D = (Operational$_D$ setParamaters$_{req}$ Operational$_D$),

τ_2^D = (Operational$_D$ observe$_{req}$ Operational$_D$),

τ_3^D = (Operational$_D$ netConnectivity$_{req}$ Operational$_D$),

τ_4^D = (Operational$_D$ cancelObservation$_{req}$ Operational$_D$),

τ_5^D = (Operational$_D$ selectBearer$_{req}$ UpdateBearer),

τ_6^D = (UpdateBearer reg$_{res}$ Operational$_D$),

τ_7^D = (Operational$_D$ T$_{mon}$ NotifyAck),

τ_8^D = (Operational$_D$ trigger NotifyAck),

τ_9^D = (NotifyAck notify$_{res}$ Operational$_D$),

τ_{10}^D = (Operational$_D$ create$_{req}$ APNconfiguration)

τ_{11}^D = (APNconfiguration reg$_{res}$ APNconfiguration)

τ_{12}^D = (APNconfiguration activateAPN$_{req}$ APNconfiguration)

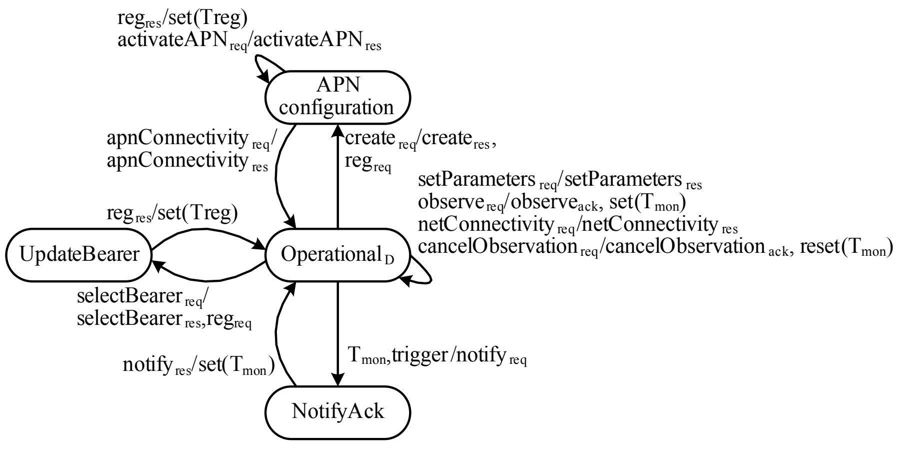

reg$_{res}$/set(Treg)
activateAPN$_{req}$/activateAPN$_{res}$
APN configuration
apnConnectivity$_{req}$/ apnConnectivity$_{res}$
create$_{req}$/create$_{res}$, reg$_{req}$
setParameters$_{req}$/setParameters$_{res}$
observe$_{req}$/observe$_{ack}$, set(T$_{mon}$)
netConnectivity$_{req}$/netConnectivity$_{res}$
cancelObservation$_{req}$/cancelObservation$_{ack}$, reset(T$_{mon}$)
reg$_{res}$/set(Treg)
UpdateBearer
Operational$_D$
selectBearer$_{req}$/ selectBearer$_{res}$,reg$_{req}$
notify$_{res}$/set(T$_{mon}$)
T$_{mon}$,trigger/notify$_{req}$
NotifyAck

Figure 2. Connectivity observation model as seen by the device

τ_{13}^D = (APNconfiguration apnConnectivity$_{req}$ Operational$_D$).

Having formal description of both state machines we can formally verify that they are synchronized.

5. A Method for Formal Verification of Connectivity Management Models

In order to prove that both LTSs are synchronized, we use the concept of weak bisimulation.

Intuitively, in terms of observed behavior, two LTSs have bisimulation relation if one LTS displays a final result and the other displays the same result [13]. In practice, strong bisimulation puts strong conditions for equivalence which are not always necessary. For example, some transitions can present actions, which are internal to the system (i.e. not observable). In weak bisimulation, internal transitions can be ignored.

Proposition. The labeled transition systems CM_S and CM_D are weakly bisimilar.

Proof. To prove that both LTSs bisimulate each other it is necessary to identify a bisimilar relation between their states.

Let U_{DS} = {(Operational$_D$, Operational$_S$), (UpdateBearer, PreferredBearerAck), (APNconfiguration, APNprofile)},

then:

1. For Operational$_D$ $\exists\{\tau_1^D, \tau_2^D\}$ and for Operational$_S$ $\exists\{\tau_1^S, \tau_2^S, \tau_3^S\}$ – setting observation policy;
2. For Operational$_D$ $\exists\{\tau_7^D, \tau_9^D\}$ and for Operational$_S$ $\exists\{\tau_4^S\}$ – periodic reporting of signal strength and line voltage;
3. For Operational$_D$ $\exists\{\tau_8^D, \tau_9^D\}$ and for Operational$_S$ $\exists\{\tau_4^S\}$ – the same as in 2 but triggered case;
4. For Operational$_D$ $\exists\{\tau_4^D\}$ and for Operational$_S$ $\exists\{\tau_{15}^S, \tau_{16}^S\}$ – observation cancelation;
5. For Operational$_D$ $\exists\{\tau_3^D\}$ and for Operational$_S$ $\exists\{\tau_5^S, \tau_6^S\}$ – network connectivity queries;
6. For Operational$_D$ $\exists\{\tau_5^D\}$ and for Operational$_S$ $\exists\{\tau_7^S\}$ – preferred bearer selection;
7. For UpdateBearer $\exists\{\tau_6^D\}$ and for PreferredBearerAck $\exists\{\tau_8^S, \tau_9^S\}$ – re-registration after preferred bearer selection;
8. For Operational$_D$ $\exists\{\tau_{10}^D\}$ and for Operational$_S$ $\exists\{\tau_{10}^S\}$ – creation a new APN profile;
9. For APNconfiguration $\exists\{\tau_{11}^D, \tau_{12}^D, \tau_{13}^D\}$ and for APNprofile $\exists\{\tau_{11}^S, \tau_{12}^S, \tau_{13}^S, \tau_{14}^S\}$ – re-registration, APN activation and checking the APN connectivity.

Therefore CM_D and CM_S are weakly bisimilar, which means that both state machines, representing the server and device views on connectivity management, are synchronized.

The proposed connectivity management models illustrate how diagnostic and monitoring information may be gathered. The usage of gathered information for QoS management is modeled in the next section.

6. Model of Autonomous QoS Management Agent

An agent is a thing that perceives from and acts on an M2M device in such way that an M2M device goes through a sequence of states maximizing the performance measures. The problem in M2M self-optimization includes a goal and set of means to achieve the goal. One of the goals of M2M self-optimization is the application of appropriate QoS to M2M connections. An agent dedicated to M2M QoS management, reasons about and follows actions in order to achieve the goal. The process of reasoning what means it can use includes decision searching. The QoS Management Agent is goal-based and solves the problem deciding what to do by finding sequences of actions that lead to the desirable QoS applied to M2M device connection. Further, we consider M2M devices with cellular or wireless connectivity. The agent actions can be viewed as transitions between states of QoS resources assigned to M2M connection.

The problem solving of an M2M QoS Management Agent includes four stages: goal formulation, problem formulation, searching solution and execution. The Agent explores the current situation and draws the goal which helps to organize behavior by rejecting actions that result in a failure to achieve the appropriate QoS. The Agent draws the problem by deciding what transitions and states to consider following the state of QoS assigned to M2M connection. In general, an M2M QoS Management Agent faces with several options of possible sequences of actions because it does not know enough about the current resource state. The Agent searches the solution space by examining different sequences of action. Once the solution is found, the agent carries out the identified actions in the execution stage.

The goal of agent for QoS manage-ment is to maintain the appropriate QoS for M2M applications. While many M2M applications can deal with best-effort QoS, some M2M applications stipulate higher QoS or priority requirements. Healthcare or supply chain applications demand more reliable connections to support life- or mission-critical applications. Bandwidth hungry applications like video surveillance, transportation services, and industrial control become a real challenge for network operators. Some M2M applications like remote control and multiuser gaming require lower latency. Allocation and retention priority, which determines the connectivity priority in case of congestion, is also important QoS requirement for some M2M applications such as transmitting seismic or heart-monitoring data.

The agent formulates the goal based on information about M2M application. The agent extracts and configures in the device specific QoS trap events that may trigger the agent activity. The problem arises when some of the traps become active which means that the predefined thresholds are reached. In the next paragraphs we present the agent's behavior in problem solving.

We use predicates to express the facts, to show the exchange of messages between the M2M device and autonomous QoS management application, and to describe the states of QoS resources assigned to M2M connection. For example the predicate usedBearer(x,b) is true if b is the used bearer for device x, and defaultQoS(x) becomes true when the requested by the agent QoS can not be applied and the default QoS is applied.

From the agent point of view the M2M device may be in one of the following states. In Null state, there are no bearer assigned and the device is not involved in session. In WaitQoSAck state, the application tries to apply specific QoS parameters based on session request and waits for acknowledgement. In WaitSubscriptionAck state, the application tries to subscribe to QoS events that may occur on M2M session and waits for acknowledgement. In AppliedQoS state, the application has applied specific QoS parameters to the M2M session. In WaitBearers state, the application waits from the device available bearers. In WaitBearerChange state, the application waits for bearer change by the device. In WaitAPN state, the application waits for APNs activated by the device. In WaitAPNactivation state, the application waits for activation of a new APN. In WaitModifyAck state the application has modified the specified QoS and waits for acknowledgement. In WaitRemoveAck state, the application has removed the specific QoS and waits for acknowledgement. In WaitStopAck state, the application terminates its subscriptions to notification about QoS events.

The behavior of the QoS Management Agent is described by temporal logic [14]. We use a minimal set of standard notations $\mathcal{G}$ for always, $\mathcal{U}$ for until, and $\mathcal{N}$ for next.

The QoS Management Agent is activated on receiving a request for establishment an M2M session:

(1) $\mathcal{G}(\mathrm{Null}(x)\rightarrow\top\,\mathcal{U}\,\mathrm{session}_{\mathrm{req}}(x))$

On receiving a request for M2M session establishment, the Agent requests setting specific QoS data on M2M session:

(2) $\mathcal{G}(\mathrm{Null}(x)\wedge\mathrm{session}_{\mathrm{req}}(x)\rightarrow \mathrm{setQoS}_{\mathrm{req}}(x)\wedge\mathcal{N}\mathrm{WaitQoSAck}(x))$

(3) $\mathcal{G}(\mathrm{WaitQoSAck}(x)\rightarrow\top\,\mathcal{U}\,\mathrm{setQoS}_{\mathrm{res}}(x))$

If the requested QoS is applied on M2M session, the Agent registers its interest in receiving notifications about QoS events:

(4) $\mathcal{G}(\mathrm{WaitQoSAck}(x)\wedge\mathrm{setQoS}_{\mathrm{res}}(x)\wedge \mathrm{requestedQoS}(x)\rightarrow\mathrm{setTimer}_{\mathrm{QoS}}(x)\wedge\ \mathrm{startQoSnotification}_{\mathrm{req}}(x)\wedge \mathcal{N}\mathrm{WaitSubscriptionAck}(x))$

(5) $\mathcal{G}(\mathrm{WaitSubscriptionAck}(x)\rightarrow \top\,\mathcal{U}\,\mathrm{startQoSnotification}_{\mathrm{res}}(x))$

(6) $\mathcal{G}(\mathrm{WaitSubscriptionAck}(x)\wedge \mathrm{startQoSnotification}_{\mathrm{res}}(x)\rightarrow\ \mathcal{N}\mathrm{AppliedQoS}(x))$

If the requested QoS can not be applied on M2M session, the Agent requests information about used and available bearers:

(7) $\mathcal{G}(\mathrm{WaitQoSAck}(x)\wedge\mathrm{setQoS}_{\mathrm{res}}(x)\wedge \mathrm{defaultQoS}(x)\rightarrow\mathrm{getBearers}_{\mathrm{req}}(x)\wedge \mathcal{N}\mathrm{WaitBearers}(x))$

(8) $\mathcal{G}(\mathrm{WaitBearers}(x)\rightarrow\top\,\mathcal{U}\,\mathrm{getBearers}_{\mathrm{res}}(x))$

If there are available bearers, the Agent makes a decision for bearer change:

(9) $\mathcal{G}(\mathrm{WaitBearers}(x)\wedge\mathrm{getBearers}_{\mathrm{res}}(x)\wedge \neg\mathrm{usedBearer}(x,b)\wedge \mathrm{availableBearer}(x,c)\rightarrow \mathrm{changeBearer}_{\mathrm{req}}(x,c)\wedge \mathcal{N}\mathrm{WaitBearerChange}(x))$

(10) $\mathcal{G}(\mathrm{WaitBearerChange}(x)\rightarrow \top\,\mathcal{U}\,\mathrm{changeBearers}_{\mathrm{res}}(x))$

If the bearer change is successful, the Agent requests information about APN profiles and activates a new one:

(11) $\mathcal{G}(\mathrm{WaitBearerChange}(x)\wedge \mathrm{changeBearers}_{\mathrm{res}}(x)\rightarrow \mathrm{getAPNs}_{\mathrm{req}}(x)\wedge\mathcal{N}\mathrm{WaitAPNs}(x))$

(12) $\mathcal{G}(\text{WaitAPNs}(x) \rightarrow \mathcal{U}\text{UgetAPNs}_{\text{res}}(x))$

(13) $\mathcal{G}(\text{WaitAPNs}(x) \wedge \text{getAPNs}_{\text{res}}(x) \rightarrow \text{activateAPN}_{\text{req}}(x) \wedge \mathcal{N}\text{WaitAPNactivation}(x))$

(14) $\mathcal{G}(\text{WaitAPNactivation}(x) \wedge \text{activateAPN}_{\text{res}}(x) \rightarrow \text{setQoS}_{\text{req}}(x) \wedge \mathcal{N}\text{WaitQoSAck}(x))$

If there are no available bearers, then no QoS resources can be assigned to the M2M session:

(15) $\mathcal{G}(\text{WaitBearers}(x) \wedge \text{getBearers}_{\text{res}}(x) \wedge \text{noAvailableBearers}(x) \wedge \rightarrow \mathcal{N}\text{Null}(x))$

During an active M2M session with applied specific QoS, notifications about QoS events may be received, or the QoS timer which determines the duration of applying specific QoS expires:

(16) $\mathcal{G}(\text{AppliedQoS}(x) \rightarrow \top \mathcal{U}(\text{notify}_{\text{req}}(x) \vee \text{Timer}_{\text{QoS}}))$

If a notification is received that all used bearers are lost, the Agent requests a bearer change:

(17) $\mathcal{G}(\text{AppliedQoS}(x) \wedge \text{notify}_{\text{req}}(x) \wedge \text{AllBearersLost}(x)) \rightarrow \text{notify}_{\text{res}}(x) \wedge \text{getBearers}_{\text{req}}(x) \wedge \mathcal{N}\text{WaitBearers}(x))$

If a notification is received that a bearer is lost, the Agent requests modification of QoS:

(18) $\mathcal{G}(\text{AppliedQoS}(x) \wedge \text{notify}_{\text{req}}(x) \wedge \text{BearerLost}(x)) \rightarrow \text{notify}_{\text{res}}(x) \wedge \text{modifyQoSs}_{\text{req}}(x) \wedge \mathcal{N}\text{WaitModifyAck}(x))$

(19) $\mathcal{G}(\text{WaitModifyAck}(x) \rightarrow \top \mathcal{U}(\text{modifyQoS}_{\text{res}}(x))$

(20) $\mathcal{G}(\text{WaitModifyAck}(x) \wedge \text{modify}_{\text{res}}(x) \wedge \text{requestedQoS}(x) \rightarrow \mathcal{N}\text{AppliedQoS}(x))$

If the QoS timer expires, the Agent removes the assigned QoS and stops notifications about QoS events:

(21) $\mathcal{G}(\text{AppliedQoS}(x) \wedge \text{Timer}_{\text{QoS}}(x)) \rightarrow \text{removeQoS}_{\text{res}}(x) \wedge \mathcal{N}\text{WaitRemoveAck}(x))$

(22) $\mathcal{G}(\text{WaitRemoveAck}(x) \rightarrow \top \mathcal{U}(\text{removeQoS}_{\text{res}}(x))$

(23) $\mathcal{G}(\text{WaitRemoveQoS}(x) \wedge \text{removeQoS}_{\text{res}}(x)) \rightarrow \text{stopQoSNotification}_{\text{req}}(x) \wedge \mathcal{N}\text{WaitStopAck}(x))$

(24) $\mathcal{G}(\text{WaitStopAck}(x) \rightarrow \top \mathcal{U}(\text{stopQoSNotification}_{\text{res}}(x))$

(25) $\mathcal{G}(\text{WaitStopAck}(x) \wedge \text{stopQoSNotification}_{\text{res}}(x)) \rightarrow \mathcal{N}\text{Null}(x))$

The functionality of the QoS Management Agent may be extended by setting usage monitoring thresholds for a group of M2M devices and configuration of specific application detection traffic and receiving notification about initiation and termination of specified application traffic.

6. Discussions

The ability to connect to and manage diverse and multiple devices in a scalable and resilient way is a challenge for network operators. A number of different kind of management platforms have been developed to support this endeavor [10,15]. Device management platforms are typically developed by a single device vendor, work across different protocols and meet specific M2M application requirements [16]. These platforms are mainly focused on cellular connectivity and support initiating, configuring and activating subscriber identity modules. Most of the existing connectivity platforms are not designed to be flexible or adaptive to different connection technologies. Once developed M2M applications, it is difficult to change devices adding connectivity technologies or integrating new application requirements with new data models. The result is multiple vertical solutions where interoperability of connectivity management capabilities is very limited or non-existent.

Our approach is based on a standar-dized framework for device manage-ment which enables development of horizontal connectivity platforms. The approach is technology agnostic and facilitates the management of multiple connectivity options. Based on high level description of OMA device management procedures, our context-aware models illustrate the way in which data captured from devices may be converted into information. Further, this information might be used to extract knowledge by modeling the autonomous agent that manages QoS available on device connections.

Such an approach removes much of the difficulty of integrating new connectivity options and opens the door to new value added services for QoS management in M2M communications.

7. Conclusions and Future Work

The LWM2M provides an open interface between the M2M device/client side and M2M service platform side which allows cost efficient remote device management. Based on the LWM2M test procedures we model the behavior of the device and server in the context of connectivity management which includes monitoring of network connectivity para-meters, bearer selection and APN configuration. We describe the models formally and prove that the models have weak bisimulation relationship which means that the models are synchronized. We illustrate the usage of connectivity management capabilities in dynamic QoS control on M2M connections by modeling an autonomous agent. The agent logic is formally described by temporal logic.

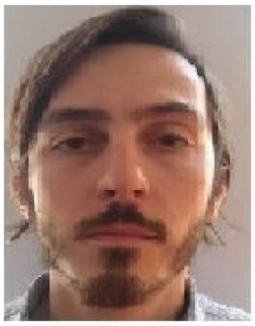
Our future work will be focused on automation of M2M registration procedures including updating of device firmware version.

References

1. Pereira, C., A. Aguiar. Towards Efficient Mobile M2M Communications: Survey and Open Challenges. – *Sensors,* 14, 2014, 19583-19608.
2. Zhang, Y., R. Yu, S. Xie, W. Yao, Y. Xiao, M. Guizani. Home M2M Networks: Architectures, Standards, and QoS Improvement. – *Communication Magazine,* 49, 2015, No. 4, 44-52.
3. Ratasuk, R., A. Prasad, Z. Li, A. Ghosh, M. Uusitalo. Recent Advancements in M2M Communications in 4G Networks and Evolution Towards 5G. IEEE International Conference on Intelligence in Next Generation Networks, 2015, 52-57.
4. Tikhvinskiy, V., G. Bochechka. Prospects and QoS Requirements in 5G. – *Journal of Telecommunications and Information Technology*, 1, 2015, 23-26. Available at: http://www.nit.eu/czasopisma/ JTIT/2015/ 1/23.pdf.
5. 3GPP TS 23.682 Architecture Enhancements to Facilitate Communications with Packet Data Networks and Applications. Release 13, 2015, 13.3.0.
6. Morrish, J. Business Models for Machine-to-machine (M2M) Communications. In Edited Book "Machine-to-Machine Communications", Elsevier, 2015, 339-353.
7. Dujak, M., Z. Kljaic, M. Serbec, H. Jelacic. Next-Generation Utilities Based on M2M Communications. International Conference on Information and Communication Technology, Electronics and Microelectronics, 2014, 507-513.
8. Klas, G., F. Rodermund, Z. Shelby, S. Akhouri, J. Hoeller. Lightweight M2M: Enabling Device Management and Applications for the Internet of Things. 2014. Available at: http://archive.erics-son.net/service/inter net/ picov/get?DocNo= 1/ 28701-FGB101973.
9. Datta, S., C. Bonnet. A Lightweight Framework for Efficient M2M Device Management in oneM2M Architecture. International Conference on Recent Advances in Internet of Things (RIoT), 2015, 1-6.
10. Corici, A. A., R. Shrestha, G. Carella, A. Elmangoush, R. Steinke, T. Magedanz. A Solution for Provisioning Reliable M2M Infrastructures Using SDN and Device Management. International Conference on Information and Communication Technology (ICoICT), 2015, 81-86.
11. Open Mobile Alliance. Diagnostics and Monitoring Trap Events Specifications 2013, OMA-TS-DiagonTrapEvents-V1_2-20131008-A.
12. Open Mobile Alliance. Enabler Test Specification for Lightweight M2M Candidate Version 1.0–03 Feb 2015, OMA-ETS-LightweightM2M-V1_0-20150203-C.
13. Liu, F., Q. Zhang, X. Chen. Bisimilarity Control of Decentralized Nondeterministic Discrete-event Systems. International Control Conference (CCC), 2014, 3898-3903.
14. Abbas, H., B. Hoxha, G. Fainekos, K. Ueda. Robustness-guided Temporal Logic Testing and Verification for Stochastic Cyber-Physical Systems. IEEE 4th Annual International Conference on Cyber Technology in Automation, Control and Intelligent Systems (CYBER), 2014, 1-6.
15. Čačković, V., Ž. Popović. Device Connection Platform for M2M Communication. International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2012, 1-7.
16. Machine Research. The Critical Role of Connectivity Platforms in M2M and IoT Application Enablement. White Paper, 2014, Available at: https://machinare search.com/news/white-paper-the-critical-role-of-connectivity-platforms-in-m2m-and-iot-application-enablement/.

Manuscript received on 08.04.2016

Ivaylo Atanasov *is with the Faculty of Telecommunications, Technical University of Sofia. He received his PhD degree in communication networks. Currently, he is professor and his scientific research area covers mobile networks, internet communications and protocols, and mobile applications.*



Anastas Nikolov *is a PhD student at the Faculty of Telecommunications, Technical University of Sofia. His doctoral thesis is in the area of autonomous management of Internet of things based systems.*

Evelina Pencheva *is with the Faculty of Telecommunications, Technical University of Sofia. She has a DSc degree in communication networks. Currently, she is professor and her scientific research area covers multimedia networks, telecommunication protocols, and service platforms.*

Contacts:
Faculty of Telecommunications
Technical University of Sofia
8 Kliment Ohridski, 1756 Sofia, Bulgaria
e-mail: enp@tu-sofia.bg