Formal Verification of Models for M2M Device Registration

Evelina Pencheva[1], Ivaylo Atanasov[1] and Anastas Nikolov[1]

Abstract – **Device management is a challenging task in Machine-to-Machine (M2M) communications. The increased number of connected devices and device diversity calls for device management that reduces the deployment time and operational costs. Open Mobile Alliance specified Lightweight M2M protocol for device management. In this paper, we propose device registration models based on Lightweight M2M procedures and a mathematical approach to behavior verification. Both server and client models are formally described, and a concept of weak bisimulation is used to prove that the models are synchronized.**

Keywords – **Device management, Finite state machines, Behavioral equivalence, Weak besimulation**

I. INTRODUCTION

Machine-to-Machine (M2M) communications have various application areas such as automotive, home automation, smart cities, energy efficiency, industry, agriculture, safety and security, health, education and others. Despite the differences all these areas set common requirements for connected devices to communicate through different access networks, remote configuration and control. Device management is a challenging and critical issue due to rapidly growing number of connected devices and their diversity [1]. A plethora of devices and customized solutions are available on the market and a large amount of the employed technology is proprietary today [2], [3], [4]. This calls for abstraction of device management functions which has to hide the complexity and to be technology independent. Such an abstraction can be provided by OMA LWM2M [5], [6]. Lightweight M2M is a protocol from the Open Mobile Alliance for M2M device management. It defines device management procedures between a LWM2M server and a LWM2M client, which is located in a device. The protocol may be used to create device management solutions that apply the approach of software defined networks [7], [8]. 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. In this paper, we suggest an approach to formal verification of LWM2M server and client behaviour related to device management. Our models are compliant with ETSI M2M functional architecture [9] and Enabler Test Specification for Lightweight M2M [10]. First we start with formal description of device registration models and using the well known concept of weak bisimulation [11] we prove formally that the models are synchronized. In addition to regular device registration functions, our models include functions related to server initiated device registration, updating the firmware version and server disabling, and prove that the models expose equivalent behaviour.

[1]The authors are with the Faculty of Telecommunications at Technical University of Sofia, 8 Kl. Ohridski Blvd, Sofia 1000, Bulgaria, E-mails: enp@tu-sofia.bg; iia@tu-sofia.bg; nikolov.anastas@gmail.com

II. DEVICE REGISTRATION MODELS

Device registration allows the server to maintain device reachability status. If the device is not registered it is not reachable. When the device sends a registration request (reg_{req}) it moves to registering state, where it awaits the server answer. After receiving the server answer with registration acknowledgement, the device sets the registration timer (Treg) and moves to registered state. If the device is registered, it is with operational firmware and the server and device store registration-related information making it available, on request or based on subscription. When the registration timer expires the device refreshes it registration. When registered, the device may receive a soft offline request and then it sends a de-registration request to the server and becomes unregistered.

The device's view point on its registration state is as follows. In $Unregistered_D$ state, the device is offline and it is not registered. In Registering state, the device is in a process of registration. In $OperationalFw_D$, the device is registered with operational firmware. In UpdateRegistration state a transport binding between the server and device is established and the device waits for registration update trigger message from the server. In WaitDeregAck state, the device waits for de-registration acknowledgement. In $FirmwareDownloading_D$ state, the device downloads the new firmware version. The model representing the device's view on its registration state is shown in Fig.1.

The server's view point on device registration state is as follows. In $Unregistered_S$ state, the device is not registered. In $OperationalFw_S$ state, the device is registered with operational firmware. In NotificationStoring state, the device is registered and the server updates the notification storing object. In Disabling state, the server will be disabled. In Transport-Binding state, the device is registered and updates the registration. In WaitUpdateAck state, the server waits for acknowledgement of transport binding. In WaitFwVersion state, the device is registered and the server reads the current firmware version. In WaitDownlodAck state, the device is registered and the server initiates the download of a new firmware version. In $FirmwareDownloading_S$ state, the device downloads the new firmware version. In WaitFwActionStatus state, the server asks for the firmware downloading status. In WaitFwUpdate state, the server waits for acknowledgement of firmware update. In RemoveOldFirmware state, the server waits for acknowledgement that the old firmware version is removed. In Rebooting state, the server waits device rebooting. The model representing the server's view on device registration state is shown in Fig.2.

III. Formal model verification

Both models are formally described as Labeled Transition Systems (LTS).

By $R_D = (S_D, Act_D, \rightarrow_D, s_0^D)$ it is denoted an LTS representing the authorized device's view on its registration state, namely:

S_D = { Unregistered$_D$, Registering, OperationalFw$_D$, UpdateRegistration, FirmwareDownloading$_D$, WaitDeregAck };

Act_D = { online, T$_{disable}$, initialReg$_{ack}$, reReg$_{ack}$, fwVersion$_{req}$, removeFw$_{req}$, updateNS$_{req}$, Treg$_D$, bind$_{req}$, updateReg$_{com}$, disable$_{req}$, reboot$_{req}$, writeFw$_{req}$, readFw$_{req}$, updateFw$_{req}$, softOffline, deReg$_{ack}$ };

$\rightarrow_D$ = { $\tau_1^D, \tau_2^D, \tau_3^D, \tau_4^D, \tau_5^D, \tau_6^D, \tau_7^D, \tau_8^D, \tau_9^D, \tau_{10}^D, \tau_{11}^D, \tau_{12}^D, \tau_{13}^D, \tau_{14}^D, \tau_{15}^D, \tau_{16}^D, \tau_{17}^D$ }

s_0^D = { Unregistered$_D$ },

where

τ_1^D = (Unregistered$_D$ online Registering)

τ_2^D = (Unregistered$_D$ T$_{disable}$ Registering)

τ_3^D = (Registering initialReg$_{ack}$ OperationalFw$_D$)

τ_4^D = (Registering reReg$_{ack}$ OperationalFw$_D$)

τ_5^D = (OperationalFw$_D$ fwVersion$_{req}$ OperationalFw$_D$)

τ_6^D = (OperationalFw$_D$ removeFw$_{req}$ OperationalFw$_D$)

τ_7^D = (OperationalFw$_D$ updateNS$_{req}$ OperationalFw$_D$)

τ_8^D = (OperationalFw$_D$ Treg$_D$ Registering)

τ_9^D = (OperationalFw$_D$ bind$_{req}$ UpdateRegistration)

τ_{10}^D = (UpdateRegistration updateReg$_{com}$ Registering)

τ_{11}^D = (OperationalFw$_D$ disable$_{req}$ Unregistered$_D$)

τ_{12}^D = (OperationalFw$_D$ reboot$_{req}$ Registering)

τ_{13}^D = (OperationalFw$_D$ writeFw$_{req}$ FirmwareDownloading$_D$)

τ_{14}^D = (FirmwareDownloading$_D$ readFw$_{req}$ FirmwareDownloading$_D$)

τ_{15}^D = (FirmwareDownloading$_D$ updateFw$_{req}$ OperationalFw$_D$)

τ_{16}^D = (OperationalFw$_D$ softOffline WaitDeregAck)

τ_{17}^D = (WaitDeregAck deReg$_{ack}$ Unregistered$_D$).

By $R_S = (S_S, Act_S, \rightarrow_S, s_0^S)$ it is denoted an LTS representing the server's view on authorized device registration state as follows:

S_S = { Unregistered$_S$, OperationalFw$_S$, TransportBinding, NotificationStoring$_S$, Disabling, WaitUpdateAck, WaitFwVersion, Rebooting FirmwareDownloading$_S$, WaitFwActionStatus, WaitFwUpdate, WaitDownloadAck, RemoveOldFirmware, };

Act_S = { initialReg$_{req}$, reReg$_{req}$, Treg$_S$, deReg$_{req}$, notifStoring, updateNS$_{ack}$, disable, fwAvailable, updateReg, disable$_{ack}$, bind$_{ack}$, updateReg$_{ack}$, fwVersion$_{res}$, writeFW$_{res}$, fwT$_{dl}$, readFw$_{res}$, updateFw$_{ack}$, remove$_{ack}$, reboot$_{ack}$ }

$\rightarrow_S$ ={ $\tau_1^S, \tau_2^S, \tau_3^S, \tau_4^S, \tau_5^S, \tau_6^S, \tau_7^S, \tau_8^S, \tau_9^S, \tau_{10}^S, \tau_{11}^S, \tau_{12}^S, \tau_{13}^S, \tau_{14}^S, \tau_{15}^S, \tau_{16}^S, \tau_{17}^S, \tau_{18}^S, \tau_{19}^S, \tau_{20}^S, \tau_{21}^S$ };

- s_0^S = { Unregistered$_S$ },

where

τ_1^S = (Unregistered$_S$ initialReg$_{req}$ OperationalFw$_S$)

τ_2^S = (Unregistered$_S$ reReg$_{req}$ OperationalFw$_S$)

τ_3^S = (OperationalFw$_S$ reReg$_{req}$ OperationalFw$_S$)

τ_4^S = (OperationalFw$_S$ Treg$_S$ Unregistered$_S$)

τ_5^S = (OperationalFw$_S$ deReg$_{req}$ Unregistered$_S$)

τ_6^S = (OperationalFw$_S$ notifStoring NotificationStoring$_S$)

τ_7^S = (NotificationStoring$_S$ updateNS$_{ack}$ OperationalFw$_S$)

τ_8^S = (OperationalFw$_S$ disable Disabling)

τ_9^S = (OperationalFw$_S$ fwAvailable WaitFwVersion)

τ_{10}^S = (OperationalFw$_S$ updateReg TransportBinding)

τ_{11}^S = (Disabling disable$_{ack}$ Unregistered$_S$)

τ_{12}^S = (TransportBinding bind$_{ack}$ WaitUpdateAck)

τ_{13}^S = (WaitUpdateAck updateReg$_{ack}$ Unregistered$_S$)

τ_{14}^S = (WaitFwVersion fwVersion$_{res}$ WaitDownloadAck)

τ_{15}^S = (WaitDownloadAck writeFW$_{res}$ FirmwareDownloading$_S$)

τ_{16}^S = (FirmwareDownloading$_S$ fwT$_{dl}$ WaitFwActionStatus)

τ_{17}^S = (WaitFwActionStatus readFw$_{res}$ FirmwareDownloading$_S$)

τ_{18}^S = (WaitFwActionStatus readFw$_{res}$ WaitFwUpdate)

τ_{19}^S = (WaitFwUpdate updateFw$_{ack}$ RemoveOldFirmware)

τ_{20}^S = (RemoveOldFirmware remove$_{ack}$ Rebooting)

τ_{21}^S = (Rebooting reboot$_{ack}$ Unregistered$_S$).

Intuitively, in terms of observed behavior, two state machines have bisimilar relation if one state machine displays a final result and the other state machine displays the same result [11]. Strong bisimilarity requires existence of homomorphism between transitions in both state machines. In practice, strong bisimilarity puts strong conditions for equivalence which are not always necessary. For example, internal transitions can present actions, which are internal to the system (i.e. not observable). In weak bisimilarity, internal transitions can be ignored. The concept of weak bisimilarity is used to study the modeling aspects of M2M device registration.

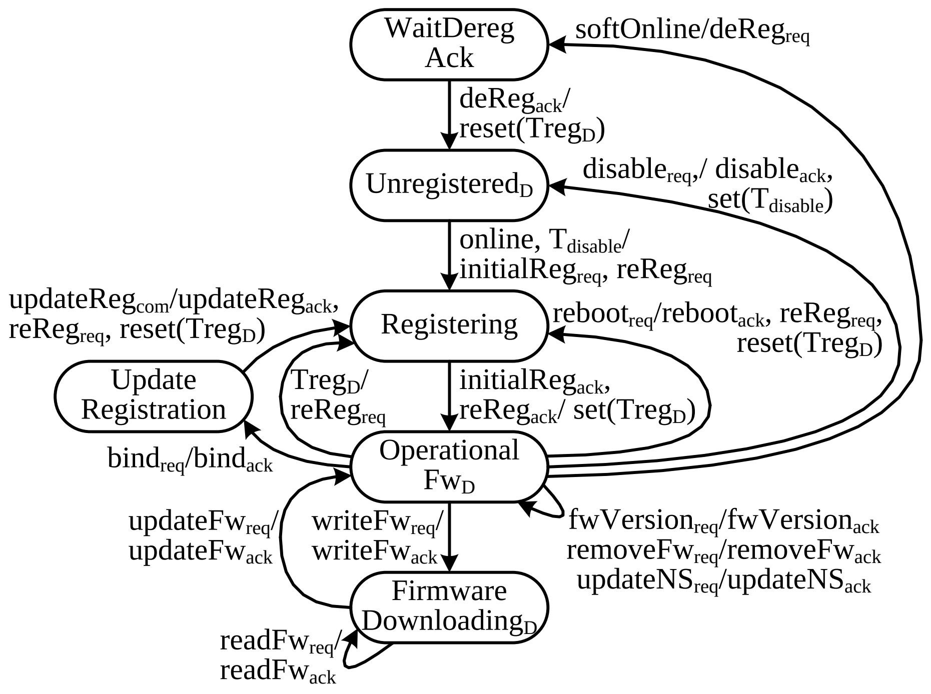
WaitDereg Ack
softOnline/deReg$_{req}$
deReg$_{ack}$/ reset(Treg$_D$)
Unregistered$_D$
disable$_{req}$,/ disable$_{ack}$, set(T$_{disable}$)
online, T$_{disable}$/ initialReg$_{req}$, reReg$_{req}$
updateReg$_{com}$/updateReg$_{ack}$, reReg$_{req}$, reset(Treg$_D$)
Registering
reboot$_{req}$/reboot$_{ack}$, reReg$_{req}$, reset(Treg$_D$)
Update Registration
Treg$_D$/ reReg$_{req}$
initialReg$_{ack}$, reReg$_{ack}$/ set(Treg$_D$)
bind$_{req}$/bind$_{ack}$
Operational Fw$_D$
updateFw$_{req}$/ updateFw$_{ack}$
writeFw$_{req}$/ writeFw$_{ack}$
fwVersion$_{req}$/fwVersion$_{ack}$
removeFw$_{req}$/removeFw$_{ack}$
updateNS$_{req}$/updateNS$_{ack}$
Firmware Downloading$_D$
readFw$_{req}$/ readFw$_{ack}$

Fig. 1. Registration state of an authorized device as seen by the device

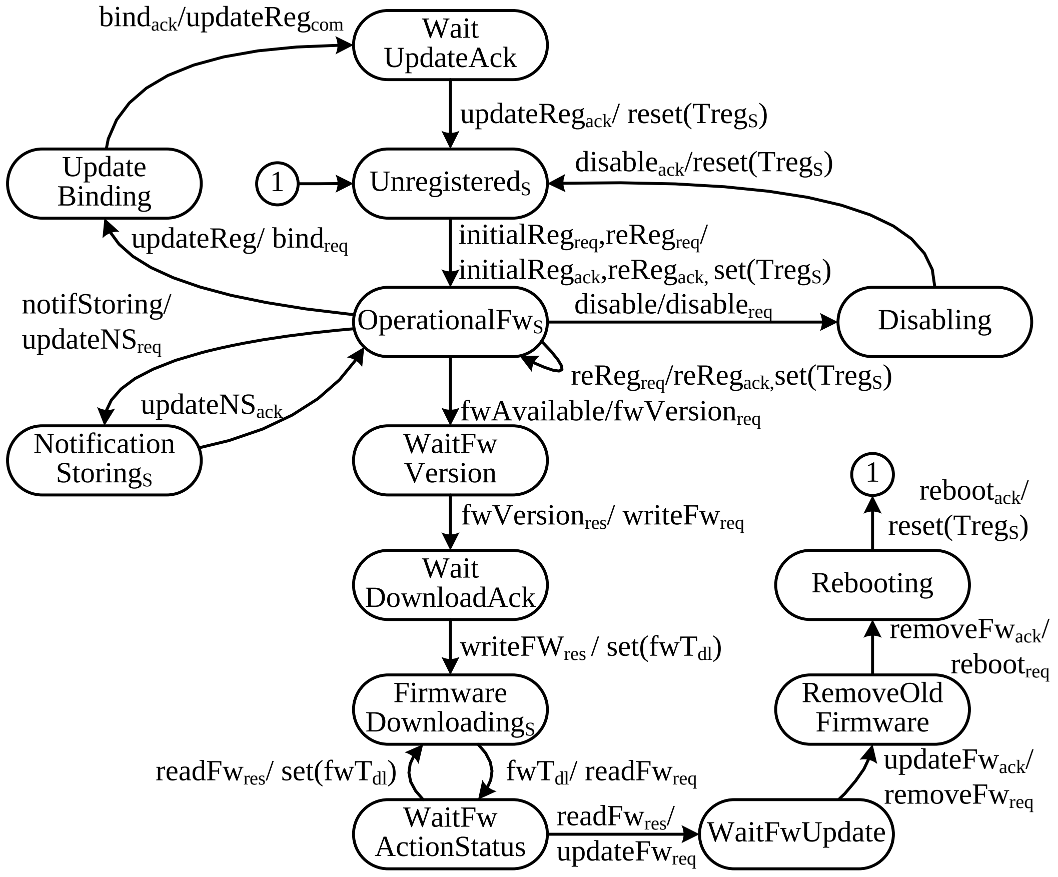
bind$_{ack}$/updateReg$_{com}$
Wait UpdateAck
updateReg$_{ack}$/ reset(Treg$_S$)
Update Binding
1
Unregistered$_S$
disable$_{ack}$/reset(Treg$_S$)
updateReg/ bind$_{req}$
initialReg$_{req}$,reReg$_{req}$/ initialReg$_{ack}$,reReg$_{ack}$, set(Treg$_S$)
notifStoring/ updateNS$_{req}$
OperationalFw$_S$
disable/disable$_{req}$
Disabling
reReg$_{req}$/reReg$_{ack}$,set(Treg$_S$)
updateNS$_{ack}$
fwAvailable/fwVersion$_{req}$
Notification Storing$_S$
WaitFw Version
1
reboot$_{ack}$/ reset(Treg$_S$)
fwVersion$_{res}$/ writeFw$_{req}$
Wait DownloadAck
Rebooting
removeFw$_{ack}$/ reboot$_{req}$
writeFW$_{res}$ / set(fwT$_{dl}$)
Firmware Downloading$_S$
RemoveOld Firmware
readFw$_{res}$/ set(fwT$_{dl}$)
fwT$_{dl}$/ readFw$_{req}$
updateFw$_{ack}$/ removeFw$_{req}$
WaitFw ActionStatus
readFw$_{res}$/ updateFw$_{req}$
WaitFwUpdate

Fig. 2. Registration state of an authorized device as seen by the server

Proposition: The labeled transition systems R_S' and R_D' are weakly bisimilar.

Proof: Let $U_{R'S'}$ = {(Unregistered$_D$, Unregistered$_S$), (OperationalFw$_D$, OperationalFw$_S$)}. Then:

1. For initial registration: Unregistered$_D$ { τ_1^D , τ_3^D } OperationalFw$_D$ ∃ Unregistered$_S$ { τ_1^S } OperationalFw$_{S;}$
2. For re-registration after offline: Unregistered$_D$ { τ_1^D , τ_4^D } OperationalFw$_D$ ∃ Unregistered$_S$ { τ_1^S } OperationalFw$_{S;}$
3. For de-registration: OperationalFw$_D$ { τ_{11}^D , τ_{17}^D } Unregistered$_D$ ∃ OperationalFw$_S$ { τ_5^S }Unregistered$_S$;
4. For re-registration due to registration lifetime is over: OperationalFw$_D$ { τ_8^D , τ_4^D } OperationalFw$_D$ ∃OperationalFw$_S${ τ_3^S , τ_4^S , τ_2^S }OperationalFw$_S$;
5. For update notification storing: OperationalFw$_D$ { τ_7^D }OperationalFw$_D$ ∃OperationalFw$_S$ { τ_6^S , τ_7^S } OperationalFw$_S$;
6. For server disabling: OperationalFw$_D$ { τ_{11}^D }Unregistered$_D$ ∃ OperationalFw$_S$ { τ_8^S , τ_{10}^S }Unregistered$_S$;
7. For re-registration when server enables: Unregistered$_D$ { τ_2^D , τ_4^D } OperationalFw$_D$ ∃ Unregistered$_S$ { τ_2^S }OperationalFw$_S$;
8. For update registration trigger: OperationalFw$_D$ { τ_9^D , τ_{10}^D , τ_4^D } OperationalFw$_D$ ∃OperationalFw$_S$ { τ_{10}^S, $\tau_{12}^S, \tau_{13}^S, \tau_2^S$ }OperationalFw$_S$;
9. For update firmware version: OperationalFw$_D$ { τ_5^D , $\tau_{13}^D, \tau_{14}^D, \tau_{15}^D$, τ_{12}^D , τ_4^D } OperationalFw$_D$ ∃OperationalFw$_S$ { τ_9^S , $\tau_{14}^S, \tau_{15}^S, \tau_{16}^S, \tau_{17}^S, \tau_{18}^S, \tau_{19}^S, \tau_{20}^S, \tau_{21}^S, \tau_2^S$ } OperationalFw$_S$.

Therefore R_S and R_D are weakly bisimilar, i.e. they expose equivalent behavior.

IV. Conclusion

The paper presents models of M2M device registration status as viewed by the server and by the device. Starting with regular models representing just registered and unregistered device state, we expand the models with additional functionality including server triggered registration update, firmware version update and server disabling. We describe models formally and prove the model synchronization by using the concept of weak bisimilarity. The models are applicable to Device Reachability, Addressing and Repository Service Capability which allows re-use in different M2M applications.

References

[1] A. Sehgal, V. Perelman, S. Kuryla, J. Schönwälder. Management of Resource Constrained Devices in the Internet of Things," IEEE Communications Magazine, December 2012, pp.144-149.

[2] V. Tayur, R. Suchithra. "Software Defined Unified Device Management for Smart Environments," International Journal of Computer Applications, 2015, vol. 121, issue 9, pp.30-34.

[3] D. Schulz, R. Gitzel, "Seamless maintenance - Integration of FDI Device Management & CMMS," IEEE Conference on Emerging Technologies & Factory Automation (ETFA), 2013, pp.402-407.

[4] C.S. Shih. C. T. Chou, K. J. Lin, B. L. Tsai, C. H Lee, D. Cheng, C. J. Chou "Out-of-Box Device Management for Large Scale Cyber-Physical Systems," IEEE International Conference on Internet of Things (iThings), and Green Computing and Communications (GreenCom), and Cyber, Physical and Social Computing (CPSCom), 2014, pp.402 – 407.

[5] V. Cackovic, Z. Popovic. "Device Connection Platform for M2M communications," IEEE International Conference on Software, Telecommunications and Computer Networks (SoftCOM), 2012, pp.1-7. Software, Telecommunications and Computer Networks (SoftCOM), 2012, pp.1-7.

[6] G. Klas, F. Rodermund, Z. Shelby, S. Akhouri, J. Höller, "Lightweight M2M: Enabling Device Management and Applications for the Internet of Things," 2014, Available at: http://archive.ericsson.net/service/internet/picov/get?DocNo= 1/28701-FGB101973.

[7] S. Datta, C. Bonnet, "Smart M2M Gateway Based Architecture for M2M Device and Endpoint Management," IEEE International Conference on Internet of Things (iThings), and Green Computing and Communications (GreenCom), and Cyber, Physical and Social Computing (CPSCom), 2014, pp.61-68.

[8] A.A. Corici, 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, pp.81-86.

[9] ETSI TS 102 690 "Machine-to-Machine communications (M2M); Functional architecture," v1.1.1, 2011.

[10] Open Mobile Alliance, "Enabler Test Specification for Lightweight M2M Candidate Version 1.0 – 03 Feb 2015," OMA-ETS-LightweightM2M-V1_0-20150203-C

[11] L. Fuchun, Z. Qiansheng, C. Xuesong, "Bisimilarity control of decentralized nondeterministic discrete-event systems," International Control Conference CCC, 2014, pp.3898-3903