2015-12

Toward enhanced data exchange capabilities for the oneM2M service platform

Glaab, M

10.1109/mcom.2015.7355583
IEEE Communications Magazine
Institute of Electrical and Electronics Engineers (IEEE)

Towards enhanced data exchange capabilities for the oneM2M Service Platform

Markus Glaab[†,‡], Woldemar Fuhrmann[†]
[†]Faculty of Computer Science
University of Applied Sciences Darmstadt
Darmstadt, Germany
markus.glaab@h-da.de

Joachim Wietzke[§]
[§]Faculty of Mechanical Engineering and Mechatronics
Karlsruhe University of Applied Sciences
Karlsruhe, Germany

Bogdan Ghita[‡]
[‡]Centre for Security, Communications and Network Research, Plymouth University
Plymouth, United Kingdom

Abstract—The Machine-to-Machine Service Platform is being standardized to enable the intercommunication of devices, which is the basis for smart environments and Intelligent Transport Systems applications. In such environments, adapting the data exchange between devices and applications to the requirements of the application is a critical step in ensuring the functionality and reliability of the service. This paper employs test-cases to analyze the data exchange of the oneM2M standard using an M2M-based Automotive Service Delivery Platform. Following the analysis, it proposes enhancements such as application-data-dependent criteria for data notification in combination with aggregation of different subscriptions to the same resource. Finally, the paper discusses the proposed enhancements against the background of M2M design considerations and improved privacy.

Keywords—Machine-to-Machine Communication; Service Delivery Platform; Automotive Software Engineering

I. INTRODUCTION

The Machine-to-Machine (M2M) service platform is being developed with the aim of overcoming existing vertical silo solutions and of building a standardized horizontal integration platform for manifold machines (also known as devices or things) and domains. This facilitates new use cases and concepts which are often referred to as 'smart' or 'intelligent', such as smart home, smart grid, smart cities, and intelligent vehicles as part of an Intelligent Transportation System (ITS).

The European Telecommunication Standards Institute (ETSI) M2M Service Architecture [1] was the first step towards a universal M2M platform, which already provided a good maturity level on unified communication capabilities that are abstracted from specific technologies and protocols. Currently, the oneM2M Global Initiative[1] works on a harmonized reference architecture, which integrates previous work such as the aforementioned by ETSI, as well as from other standardizing organizations, for example, Open Mobile Alliance (OMA) and Broad Band Forum (BBF). OneM2M released the first version of their service platform specification in April 2015 [2]. Future releases are expected to feature full semantic interoperability, which is necessary to facilitate inter-vendor and inter-domain applications without additional a-priori agreements [3].

From the software engineering perspective, M2M use cases consist of distributed inter-connected applications on top of the oneM2M service platform. This means, prior to the data processing within an application on a device or server, input data must be acquired from other components such as measurements from a large number of sensors, or preprocessed content from other machines. Therefore, having adequate data exchange capabilities is a key factor for the oneM2M service platform because it affects the service quality during operation. In particular for devices connected via a wireless access network, the constraints of the respective transport networks have to be reflected. Herein, the ability to tailor data acquisition to the requirements of a particular use case directly impacts on resulting bandwidth requirements, which may conflict with the capabilities of the transport network. This is a deciding factor as to whether a particular

[1] http://www.onem2m.org

functional split between applications and devices is feasible, including the respective costs.

This paper focuses on the data exchange capabilities of the current oneM2M specification and introduces a series of enhancements to support application-data-dependent notification criteria including local aggregation. This can enable significant bandwidth saving for many distributed usage scenarios. The considerations arose in the context of research on the applicability of M2M as the enabler for future distributed automotive software platforms [4]. The paper starts by introducing the developed architecture of an M2M-based Automotive Service Delivery Platform (ASDP) and two exemplary use cases in Section two. Section three presents the analysis of current data exchange capabilities and proposes enhancements derived from a typical automotive scenario. Section four details a number of enhancements, their implementation and its evaluation against the introduced automotive scenario. Section five discusses the approach and limitations within the wider context of oneM2M standardization, and sketches future work. Finally, Section six summarizes the contributions of this study.

II. An M2M-based Automotive Service Delivery Platform: Background, Architecture, and use cases

This section provides the foundation for an M2M-based ASDP. It starts with a short introduction to the automotive software domain. Afterwards, the architecture and two use case examples are described.

A. Background

The automotive domain is changing. More than 80% of the vehicular innovations are related to electronics and software [5]. This is driven by several factors: In-Vehicle Infotainment (IVI) systems must compete with developments in the area of consumer electronics (CE) products, in particular smartphones and tablets. Besides, Advanced Driver Assistance Systems (ADAS) aim to continuously increase the traffic safety towards zero accident through assistance and (semi-) autonomous driving capabilities. Finally, within the superior vision of ITS, vehicles are getting an integral part of our connected world, aiming a further increase of traffic efficiency, safety, and comfort of its users [6].

Hence, the expectation is that future vehicles will be connected to the Internet and to neighboring peer vehicles and infrastructure. In this regard, the automotive domain is a prime example for the concept of an Internet of Things (IoT) and its inherent challenges. Car manufacturers or suppliers – in the following referred to as Original Equipment Manufacturers (OEM) – are now facing the task of integrating applications and services from several platforms and domains. These applications differ in many ways, such as innovation- and lifecycles, performance and real-time requirements, and criticality [7]. However, the OEMs have to integrate them to a homogeneous overall system that remains functional over the complete lifetime of the car [8]. This requires new automotive software architectures that are able to handle the heterogeneity of future vehicular application landscape [5], [9].

B. Architecture

Connected vehicles raise new challenges for current software development but they also enable new ways to address them. One promising approach for the software architecture of the next generation of automotive applications is an ASDP [4], aiming the increased utilization of connectivity together with server infrastructure. Thereby the hub for the integration of automotive applications is shifted from the vehicle to a related OEM server. The offloading of existing automotive applications and the cloud-based implementation and integration of new OEM and third party functionality typically face less computational constraints than the traditional approach of the integration on automotive embedded systems. Further, an intermediary OEM server between the vehicle and third party applications and domains increases the mediation capabilities. Thus, the new approach is advantageous for many applications, in particular those that require in any case connectivity [10].

Some manufacturers have already started to build proprietary cloud solutions. However, in our ASDP approach, we envisage the alternative of an open and standardized architecture, not limited to one vendor or the automotive domain. Further, a more extensive network-integration of vehicles is intended, towards an ‘Embedded Internet’ [11]. In this regard, current developments within M2M architectures suit well the approach of an ASDP, and M2M has been selected as the underlying platform [10].

Following the concepts of the oneM2M service platform [2], a vehicle could be an Application Service Node (*ASN*) or a Middle Node (*MN*) that integrates all vehicle-internal *ASN*s or Non-oneM2M

Device Nodes (NoDN). Within our ASDP, it has been decided that the vehicle is an oneM2M-compliant *ASN*, and the OEM server is an Infrastructure Node (*IN*). The *ASN* is located inside the field domain and it is connected to the infrastructure domain, using wireless access networks.

The *ASN* and *IN* are divided into the application layer and the Common Services Entity (*CSE*), with the aspiration to encapsulate essential functions for M2M Application Entities (*AE*). This reflects the objective of a universal, horizontal integration platform.

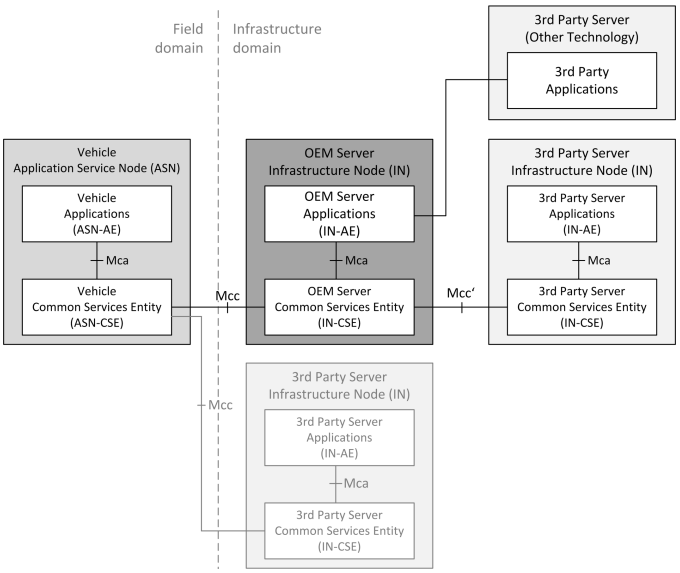

Fig. 1. Functional Architecture of an M2M-based Automotive Service Delivery Platform

Fig. 1 illustrates the compound functional architecture of an M2M-based ASDP with the reference points: *Mca* (vertical interface for *AE*), *Mcc* (horizontal between two *CSEs*). oneM2M-compliant third party servers are connected by use of the *Mcc'* interface with the OEM server, while other third party platforms could be connected through adaptor-*AE*s. For the sake of completeness, the possibility of a direct connection between the vehicle and other M2M-conformant *IN*, e.g., third party servers, via the *Mcc* interface is hinted. Against the background of mediation capabilities between a vehicle and third party server, the ASDP concept describes the value of an interposed OEM server. Hence the direct connection to third party servers is currently not favored.

C. Use cases

The ASDP concept with an OEM server as hub for application integration facilitates many use cases, currently associated with connected vehicles. To evaluate the data exchange capabilities of oneM2M, this section introduces two basic use cases, which are widely accepted.

1) Extended Floating Car Data

With Extended Floating Car Data (XFCD) [12], vehicles are used as driving traffic information sensors. They periodically report at a minimum their current location, together with the timestamp to the OEM server. The trigger for these reports may be time-related, distance-related, or a combination of both. The OEM server, respectively a third party traffic management center, aggregates and analyses the data and traffic models and can then detect traffic jams and calculate average trip times. This information can be used by navigation systems that are able to consider the current traffic situation. XFCD includes the transmission of additional sensor data to the OEM server, if it detects critical situations through the vehicular sensors. Triggers may vary from outside temperature or rain intensity to driving dynamic control interventions, such as an Electronic Stability Control (ESP) intervention. The provision of these measurements, together with the position and speed, enables advanced inference regarding the momentary traffic safety conditions on a certain road section.

2) Vehicle Maintenance / Fleet Management

Modern vehicles have variable service intervals, depending on their usage, which is monitored over time to estimate when thresholds are exceeded and service has become necessary. Additionally, various vehicular sensors and check routines continuously monitor component status and individual component failures. These are currently only locally stored using a fault recorder and manually readout at the garage. Connected vehicles enable use cases, where relevant data can be submitted to the OEM server periodically, or upon error occurrence. The gathered data may be subsequently used to initiate a separate business process of contacting the vehicle owner, discuss necessary service amounts, arrange workshop dates, or, in a wider scope, it might be used for quality management and product improvements. Remaining fuel range might also be monitored, to trigger other use cases that may propose a cheap gas station on the route.

III. Analysis of current data exchange capabilities of the oneM2M Service Platform

The following section provides an overview of the current capabilities for data exchange, based on a selected ASDP scenario, and then follows up with a discussion on recommended enhancements.

A. Principles

At the core of oneM2M interworking between *AE*s and *Node*s is a generic Resource Tree (RT), located inside each *CSE*. In addition to the structured storage of application data (such as sensor measurement), the RT facilitates essential functions, such as, registration, discovery, deregistration, announcement, grouping, subscription and notification management. From an implementation perspective, the RT is mapped to Uniform Resource Identifiers (URI) and exposed through the standardized interfaces (*Mca*, *Mcc*, *Mcc'*), following the RESTful architectural style. Accordingly, the resources are manipulated using Create, Retrieve, Update, Delete plus Notify methods (CRUD+N) [2], which are mapped to the applied application layer protocols, most likely Hypertext Transfer Protocol (HTTP), Constrained Application Protocol (CoAP) [13], or Message Queue Telemetry Transport (MQTT)[2].

According to the middleware approach of oneM2M, *AE*s exchange application data using the capabilities of the *CSE*(s), offered through the standardized interfaces. At first, the *AE* stores application data in the RT structure of their local or a remote *CSE*. For this it uses a *container* resource, which can contain, besides others, one or many *contentInstances*, according to specifiable memory constraints, such as *maxNrOfInstances*, *maxByteSize*, and *maxInstanceAge*. The actual application data is then stored within the attribute *content* of a *contentInstance*. The stored application data can be received from other *AE*s in various ways. For instance, other *AE*s can Retrieve this data on demand. Optionally, to Retrieve certain resources or subsets, conditions could be defined that are related, e.g., to time, size, state, label, number of matches, or content type of the resource.

In addition, oneM2M provides a subscribe/notify mechanism, where *AE*s can subscribe to resource changes. Similar to the Retrieve method, constraints can also be defined for the Notification, by time, state, size, or status. Furthermore, the subscribe/notify mechanism can include communication-related constraints, from batch notification, rate limit, or priority, to comprehensive notification schedule policies. These capabilities can be used to improve the 'network friendliness' of M2M traffic.

Since the subscribe/notify data exchange mechanism provides significant time and space decoupling of *AE*s, it is particularly suitable for distributed M2M use cases across different devices, vendors, and domains and is therefore selected for the ASDP use cases.

B. Analysis

In the following, a simplified scenario derived from the two use cases with one vehicle and one OEM server *Node*, and three *AE*s (see Fig. 2) is used for analysis of current data exchange capabilities of oneM2M. Since vehicular sensor data is the foundation for many automotive-related applications, it is made available to all *AE*s within the ASDP. Accordingly, an AE_1 'Vehicle Data Provider' was introduced to exemplarily make input data, such as position (latitude, longitude, heading) speed, and ESP control intervention info available in the local *CSE* RT through appropriately structured *container* resources. The AE_1 proprietarily obtains the vehicle data from an external source, such as a Controller Area Network fieldbus (CAN-bus), see step 1. Position data is usually determined using an internal Global Positioning System (GPS) receiver, connected to the CAN-bus with a typical update rate of 1 to 4 Hz, while other sensor values might be available at a much higher rate. In step 2 the AE_1 pushes the data – unaware of requirements of (future) *AE*s within the ASDP – with undiminished resolution to the *container* VehicleData, where it is stored within the attribute *content* of a *contentInstance* resource. The AE_1 additionally can specify a *contentInfo*, which is a composite attribute of an Internet Media Type and encoding information, e.g. base64 encoded string. The *CSE* further expands the *contentInstance* with the

[2] http://www.mqtt.org

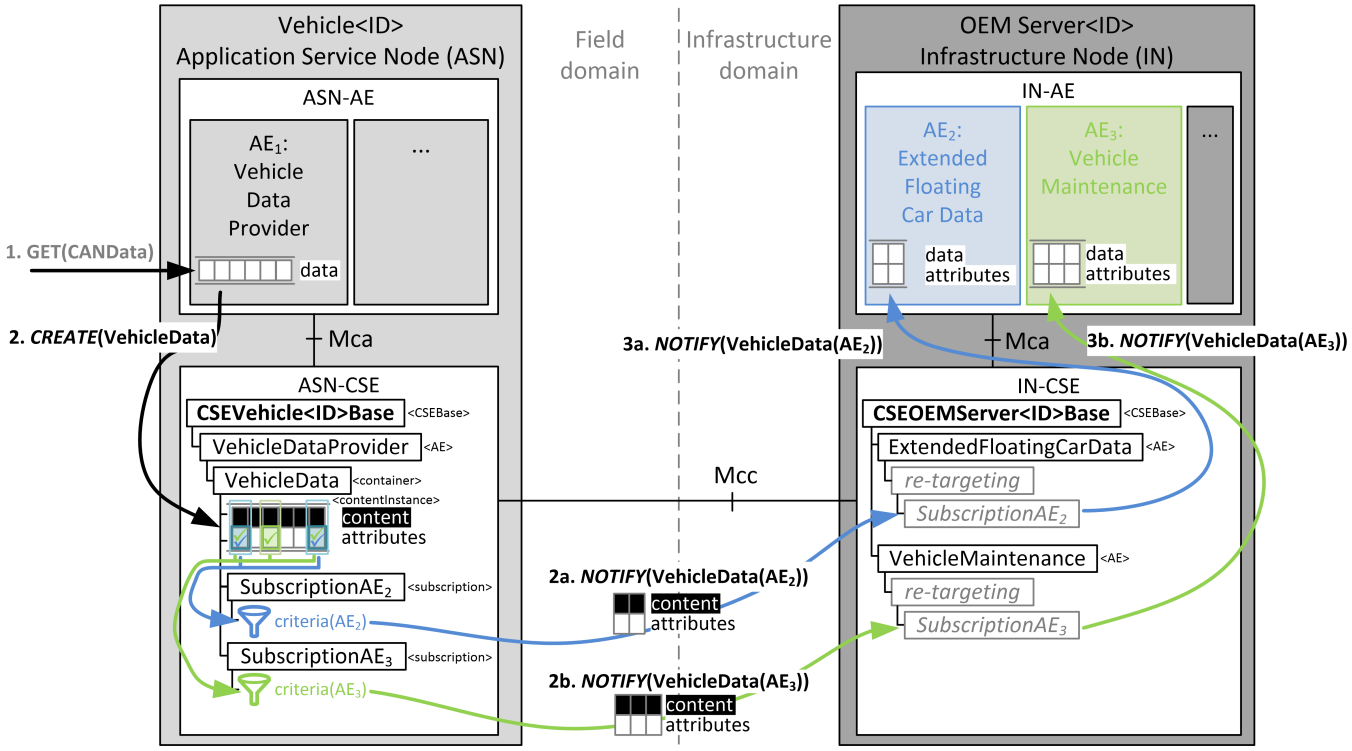

Fig. 2 An automotive scenario with current M2M data exchange capabilities

typical resource attributes, such as *contentSize* and *creationTime*.

The scenario assumes the existence of two *AE*s at the *IN*: AE_2 'Extended Floating Car Data', and an AE_3 'Vehicle Maintenance', which implement the respective application logic. Both *AE*s in this scenario subscribe to the VehicleData of the *ASN* with appropriate notification criteria. According to the existing oneM2M capabilities, time-related *schedule*s are defined: AE_3 configures the *scheduleElement*, for example, to receive notifications with the *latest* representation of the *content* resource at maximum every 5 seconds, AE_2 configures 10 seconds. These subscriptions are sent through the local *IN-CSE*, where they are both re-targeted to the target *ASN-CSE*. For the opposite notifications, local callbacks (within the *IN-CSE*) are created to route them to the *AE*s. (These steps are by-passed in Fig. 2). In the given example, the criteria(AE_2) according to its configuration selects two *contentInstance* resources out of six, hence the *ASN-CSE* performs two Notify operations that contain the *latest* representation of the resource including the VehicleData *content* (Step 2a). Similarly, the criteria(AE_3) selects three *contentInstance* resources. Thus, the *ASN-CSE* performs three notifications that contain the *latest* representation of the resource (Step 2b). Both are in each case re-targeted at the *IN-CSE* to their originators AE_2 (Step 3a) and AE_3 (Step 3b).

The example reveals two drawbacks of current oneM2M data exchange capabilities:

1) No aggregation of subscriptions

Subscriptions are not aggregated or harmonized at the local or transit *CSE*(s). This potentially leads to redundant data transmissions as the example, illustrated in Fig. 2, shows: Here, five notification messages with the respective *latest* representation of the VehicleData *container* are transmitted from the vehicle to the OEM server, whereas only three of these represent different *contentInstances* – two are redundant.

2) No application-data-dependent criteria for notification

By design-choice of the current oneM2M releases, the *contentInfo* attribute is not used at *CSE* level. Accordingly, the application data (stored within the attribute *content* of the *contentInstance* resource) becomes opaque. This has a significant impact on the available notification criteria for the subscribe/notify mechanism, i.e. the *eventNotificationCriteria* conditions. These can currently not refer to the *content*, but only can refer to other attributes of the *contentInstance* resource. For example, *eventNotificationCriteria* can relate to the *creationTime* of the *contentInstance* (with *createdBefore*, *createdAfter* conditions), or to the *contentSize* (with *sizeAbove*, *sizeBelow* conditions). Application-data-dependent notification criteria that are derived from the real use case requirements, tailoring the transmitted data to the actual needs, cannot be applied.

In the absence of such criteria, it has to be assumed that the inaccurately selected data transmitted needs to be further filtered at the receiving *AE*. Hence, it is very likely that the network bandwidth consumption of the distributed M2M applications is above the effective requirements of the use case.

Finally, the opaque *content* prevents the specification and detection of application-data-dependent events for notification. This, for instance, allows a short ESP intervention, which is only reflected within the opaque structure of a related *content*, to be missed, unless this value is provided within a separate container resource (on which an 'on change' subscription is sufficient). However, the detection of an application-data-dependent threshold exceedance is still not possible.

IV. PROPOSED ENHANCEMENTS FOR DATA EXCHANGE CAPABILITIES OF THE ONEM2M SERVICE PLATFORM

This section describes the proposed enhancements for data exchange capabilities of oneM2M. It starts with an overview of its objectives, used to derive a set of requirements followed by a description of a potential implementation. The discussion concludes with an evaluation on the basis of the introduced scenario.

A. Objectives and requirements

For the enhancements of oneM2M standards, we propose to include the following capabilities:

1) Subscription aggregation

- Different subscriptions to the same remote resource shall be aggregated at the local *CSE*.
- Different subscriptions to the same remote resource should be aggregated at transit *CSE*(s).

2) Enhanced notification

- Applications shall provide their application data in a standardized way that enables application-data-dependent notification criteria for subscribe/notify mechanism.
- A comprehensive language shall be provided to enable the standardized description of gainful application-data-dependent notification criteria including basic arithmetic and logical operations on common data types.

The left column of Table 1 names some advantageous automotive notification criteria with respect to the introduced use cases.

TABLE I. EXAMPLES OF AUTOMOTIVE APPLICATION-DATA-DEPENDENT NOTIFICATION CRITERIA AND THEIR EPL STATEMENT REPRESENTATION.

Description	EPL Statement
Notification, if remaining fuel range is smaller than 100 km.	SELECT * FROM VehicleData WHERE fuelRange < 100
Notification, if heavy rain is detected.	SELECT * FROM VehicleEnvironmentData WHERE rainSensor > 4
Notification, if there is a risk of freezing rain.	SELECT * FROM VehicleEnvironmentData WHERE temperature < 3 AND rainSensor > 0
Notification, if there is a electronic stability control intervention.	SELECT * FROM VehicleData WHERE ESP=true
Notification, if a strong deceleration of greater than 6m/s^2 is detected (calculated on speed delta [km/h] and time delta [s]).	SELECT * FROM pattern[a=Position -> b=Position((b.speed - a.speed)/((b.timestamp-a.timestamp)*3.6) < -6)]

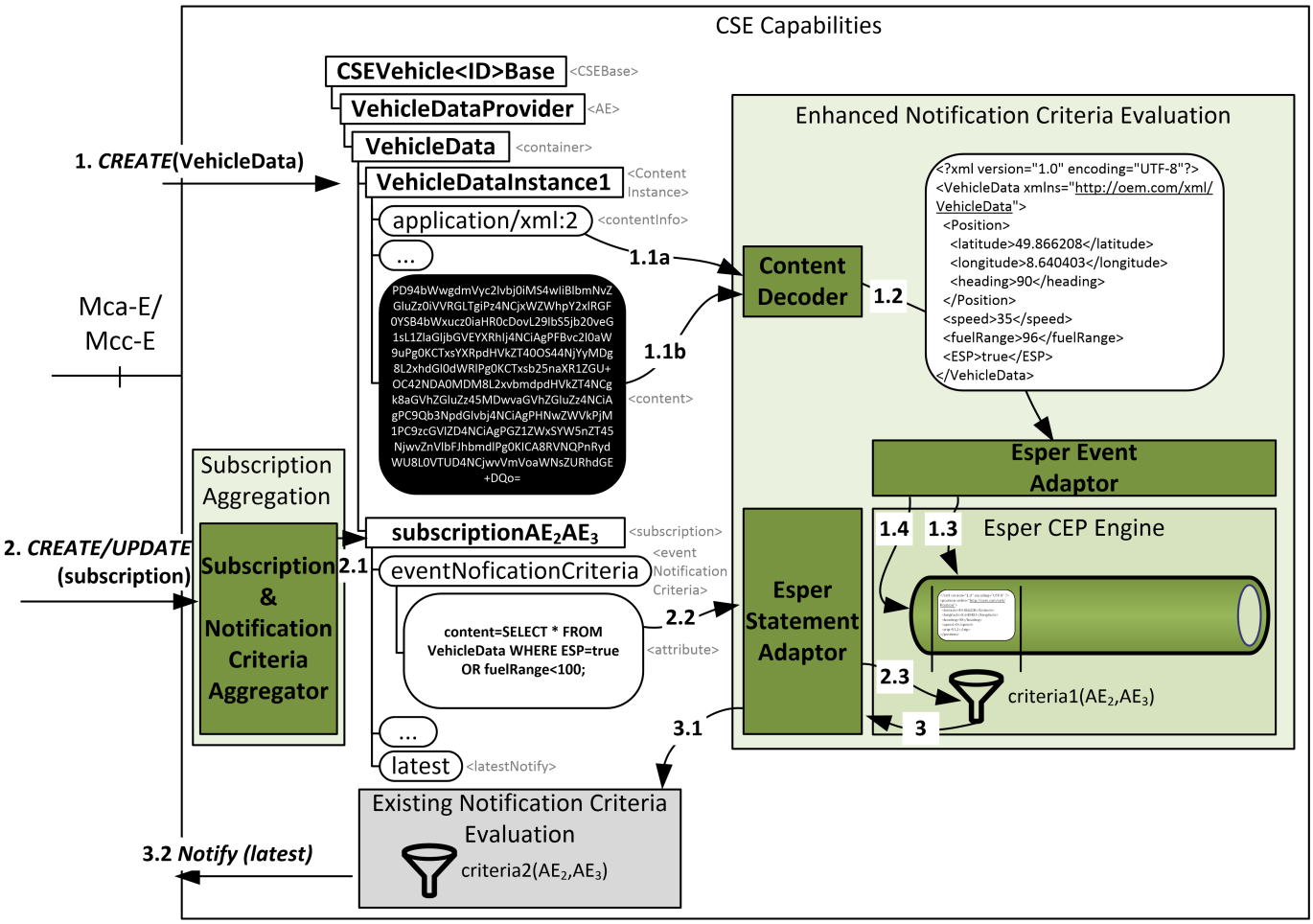

Fig. 3 Implementation of application-data-dependent notification criteria within the CSE

B. Implementation

The applicability of the proposed enhancements was validated by means of a prototype implementation. The eclipse OM2M project[3] was used for basic functionality of the oneM2M service platform. Comprehensive capabilities for the continuous analysis of data streams are available within the field of Complex Event Processing (CEP) [14]. In contrast to other approaches of CEP for M2M, such as [15], which focus on the analysis of machine-generated data at a server, we use CEP mechanisms at each M2M node to analyze (and filter) the data at its source and tailor the data transmitted. We have integrated the open-source Java Esper CEP[4] component into OM2M to add enhanced data stream processing capabilities to the *CSE* layer. Fig. 3 illustrates the prototype and the interaction of enhancements with existing CSE capabilities within the given scenario.

The starting point is the creation of a new *contentInstance* VehicleDataInstance1 within the *container* VehicleData (step 1). If *contentInfo* is set to 'application/xml:2' (which indicates a base64 encoded string of an XML document), this encoding information is forwarded to a content decoder (step 1.1a) together with the related content (step 1.1b) and then, in step 1.2, passed to the Esper event adaptor. The XML document can now be verified against the included link to at least one XML Schema Definition (XSD), in this example 'http://oem.com/xml/VehicleData', which is therefore downloaded. If the Esper event adaptor retrieves an XSD file for the first time, this is passed to the Esper CEP Engine to register an associated event type (step 1.3). Afterwards, in step 1.4 the content XML is sent to Esper as a new event.

[3] http://www.eclipse.org/om2m/

[4] http://www.espertech.com/esper/

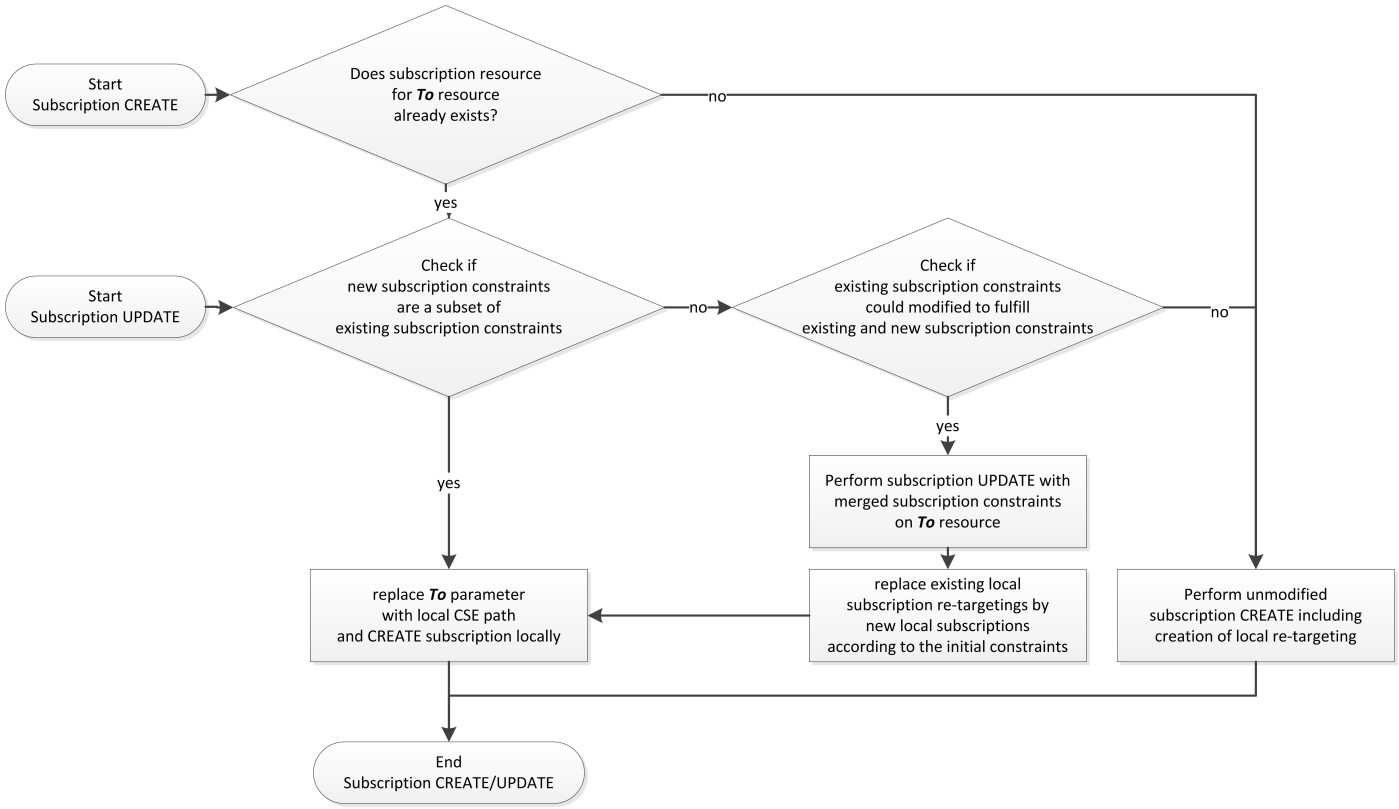

Fig. 4 Flowchart for aggregation of subscription constraints at local CSE.

The second part of the enhancements refers to the subscription Create/Update (step 2). Prior to adding the subscription to the remote *container* resource, it is aggregated with existing ones to the same resource at the local *CSE*, as illustrated in Fig. 4. For the description of application-data-dependent *eventNotificationCriteria*, the Esper Event Processing Language (EPL) is used, whose semantic and syntax is close to the Structured Query Language (SQL). The EPL statements to trigger the notification are enclosed in the *attribute* condition tag, referred to the content. Fig. 3 shows the aggregated application-data-dependent EPL statement that facilitates both, AE_2 and AE_3 criteria. The overall *eventNotificationCriteria* consist of the application-data-dependent part criteria1(AE_2,AE_3) and may have a second part criteria2(AE_2,AE_3), related to the existing notification criteria on other *contentInstance* attributes. The EPL statement is extracted by the Esper Statement Adaptor and added to the Esper CEP Engine (steps 2.2 and 2.3).

If a new *contentInstance* is created and a related subscription that includes an EPL statement exists, the *content* is forwarded to the Esper CEP Engine (steps 1.1a, 1.1b, 1.2). If the EPL statement (criteria1) is fulfilled, the continuation of notification criteria evaluation (criteria2) is triggered (step 3.1). If these are also fulfilled, a Notify is sent according to the subscription (step 3.2).

The enhancements were implemented aiming to have a minimal modification on the reference points. But, to provide the enhancements to the *AE*s and *CSE*s, the reference points had to be enhanced, which is indicated through the adapted naming *-E. Nevertheless, the application-data-dependent notification criteria remain optional and *AE*s can still provide opaque *content* that is transferred according to existing capabilities.

C. Evaluation

Fig. 5 illustrates how the proposed enhancements positively affect the scenario of Fig. 2. It is assumed that initially the AE_3 creates a subscription with criteria(AE_3) on the VehicleData *container*. According to the new capabilities, a criteria derived from the use case 'Extended Floating Car Data' is used instead of an unspecific time constraint. In this example, this criteria is the interference of the ESP. Afterwards, the AE_2 creates a subscription also to the VehicleData with the criteria(AE_2) that requests

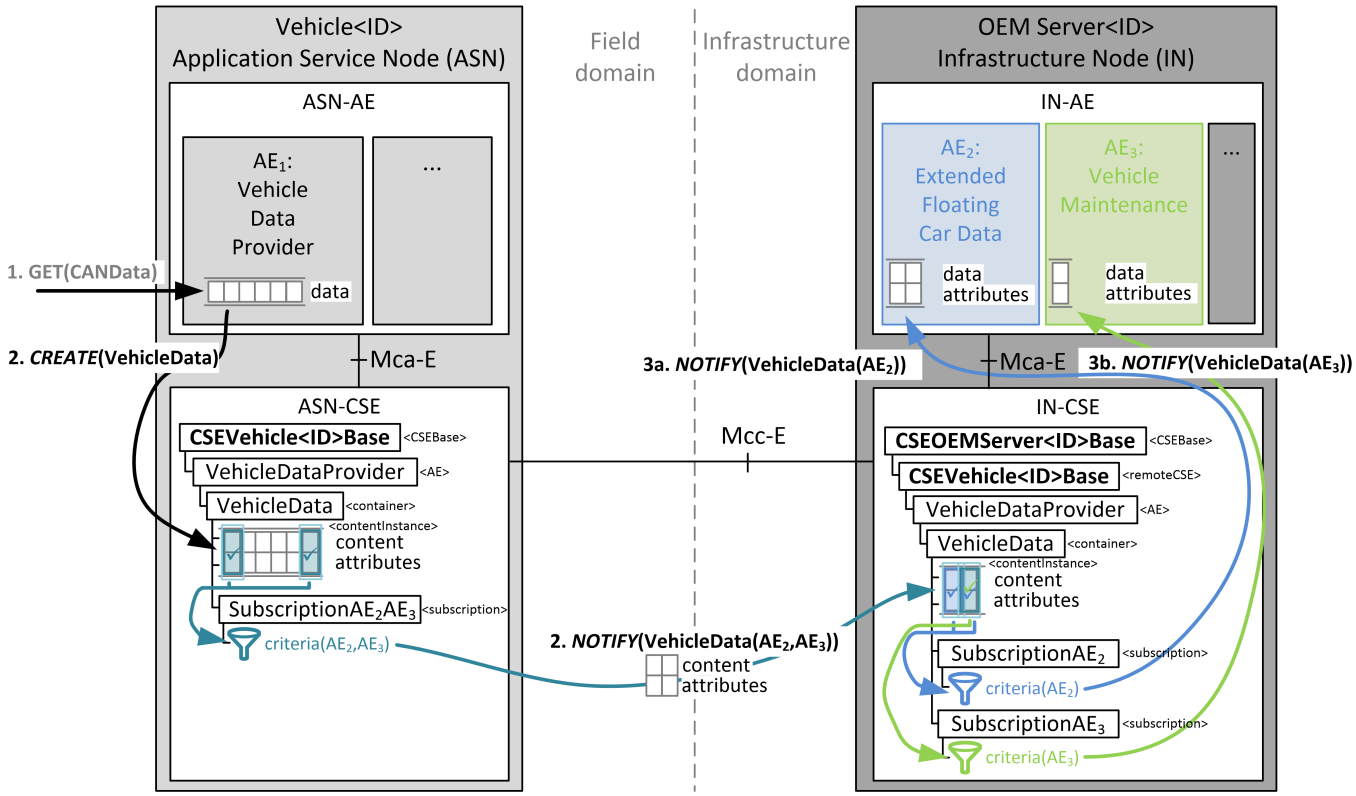

Fig. 5 An automotive scenario with enhanced M2M data exchange capabilities

notifications, if the remaining fuel range is below 100 km. At the time of the second subscription Create, the local *CSE* detects that a subscription to the same remote resource already exists and aggregates the two criteria (denoted criteria(AE_2,AE_3)). Fig. 3 shows the resulting aggregated EPL statement. Other EPL examples are listed at the right column of Table 1.

The example illustrated in Fig. 5 assumes that the ESP intervention is true at the first and the last *contentInstance*. It further assumes that the remaining fuel range at the last *contentInstance* is first time below 100. This leads to a total of two notifications, which are further distributed at the *IN-CSE*. The AE_2 receives two notifications (including respective *latest contentInstance*), the AE_3 receives one notification.

V. Discussion

The development of an M2M service platform is a trade-off between a too application-specific or domain-specific platform (which is just another silo solution) and a too common platform with too little capabilities, which might be inefficient and again causes silos – the applications on top. In this regard, any (additional) functionality of the CSE could be controversial.

The existing data exchange capabilities of oneM2M might be sufficient for scenarios, where limited sensors provide their measurements at low frequency (hence have low bandwidth consumption). They are also appropriate, when the whole range of data should be acquired, for example, in the context of big data, where data analysis is performed later offline.

Although not limited to, our proposed enhancements are particularly beneficial for domains such as automotive, where sensors or nodes possibly provide a high amount of data, of which only certain subsets are required for a use case. In such a scenario, it is neither feasible nor reasonable to transfer all sensor measurements with respect to the large number of vehicles and resulting bandwidth requirements to wireless access networks. The a priori clipping of sensor data available within the oneM2M service platform is not a suitable solution, since the car manufacturer hardly can estimate future use cases and related data requirements. Additionally, they could possibly come from different vendors and domains. Here, as indicated, application-data-dependent notification criteria together with local aggregation of subscriptions enable significant

bandwidth savings for many use-cases and may make certain functional splits between distributed applications possible at all.

The capability to detect application-data-dependent events at CSE level supports oneM2M use cases, related for example to distribute control system applications.

The proposed enhancements of application-data-dependent notification criteria might also improve privacy. On one hand, the capability to better tailor the data acquisition to the actual use case requirements can prevent applications from receiving more data than necessary, only due to the lack of appropriate notification criteria capabilities within the *CSE*. On the other hand, it could be assumed that a binary decision whether an *AE* is or is not allowed to perform a certain CRUD+N operation on a resource will no longer be sufficient. In our opinion, applications may require more detailed access specifications. In this regard, transparent *content* structures at *CSE* level could facilitate future oneM2M enhancements towards application-data-dependent *accessControlPolicies*, similar to the proposed enhancements for notification criteria. For example, enhanced access policies could prohibit applications to:

- Create subscriptions with notification criteria that analyze the vehicle speed.
- Subscribe to position updates of the vehicle with an update interval smaller than every 15 minutes.
- Use the most accurate vehicle position data available.

This, in turn, could enable further differentiation between applications and groups, such as OEM applications, third party applications, safety-related applications, or consumer applications. If users are empowered to configure such enhanced access policies for certain applications, for instance through mechanisms of the ASDP, this could finally be one aspect to better address the 'right to privacy' or improve 'informational self-determination'.

Our enhancements at this time only use a subset of the overall CEP possibilities. Similarly to the existing oneM2M capabilities, so far the enhanced notification criteria are also limited to the respective resource, where the subscription is placed. This means that, criteria across several *containers* are currently not supported. Furthermore, the notification *content* is currently not configurable, which offers possibilities for future enhancements. In this context, ongoing oneM2M standardization activities towards full semantic interoperability are beneficial, too; virtual resources, dynamically created according to the requirements of an *AE* through semantic mash-up, could be one solution to the aforementioned limitations.

VI. SUMMARY

The oneM2M service platform is currently developed as the standardized horizontal enabling platform for smart homes, smart cities, and intelligent transportation systems, all implemented as distributed inter-connected applications. In this context, data exchange capabilities represent a key functionality with respect to network efficiency and possible functional splits between applications and nodes.

This paper presented an analysis of data exchange capabilities of current oneM2M service platform specification by means of vehicular use cases, which are implemented with an M2M-based Automotive Service Delivery Platform. We found that the absence of local aggregation of different subscriptions to the same resource can cause redundant data transmissions. Besides, the opaque handling of application data at the Common Service Entity prevents the definition of application-data-dependent notification criteria for the subscribe/notify mechanism. This reduces the capabilities to tailor the data acquisition to the actual requirements of the use case. Furthermore, the detection of application-data-dependent events, such as a threshold exceedance, cannot be used as notification trigger.

Our proposed enhancements use existing oneM2M attributes, available for technical interworking of different oneM2M applications, to decode the opaque application data at the Common Service Entity level. The introduction of a Complex Event Processing engine facilitates the specification of complex statements, which enable application-data-dependent criteria for notifications including the detection of events. This enables significant bandwidth savings for many oneM2M usage scenarios, not limited to the automotive domain.

VII. REFERENCES

[1] ETSI, "Machine-to-Machine communications (M2M); Functional architecture," TS 102 690, V.2.1.1, Oct. 2013.

[2] oneM2M, "Functional Architecture," TS-0001, V.1.6.1, Jan. 2015.

[3] oneM2M, “Study of Abstraction and Semantics Enablements,” TR-0007, V.2.5.1, Jul. 2015.

[4] M. Glaab, W. Fuhrmann, J. Wietzke, and B. Ghita, “A M2M-based Automotive Service Delivery Platform for Distributed Vehicular Applications,” in *Proc. 10th Int. Network Conference*, Plymouth, UK, 2014, pp. 35–45.

[5] M. Broy, I. H. Kruger, A. Pretschner, and C. Salzmann, “Engineering Automotive Software,” *Proc. of the IEEE*, vol. 95, no. 2, pp. 356–373, Feb. 2007.

[6] T. Kosch, I. Kulp, M. Bechler, M. Strassberger, B. Weyl, and R. Lasowski, “Communication architecture for cooperative systems in Europe,” *IEEE Commun. Mag.*, vol. 47, no. 5, pp. 116–125, May 2009.

[7] M. Broy, “Challenges in automotive software engineering,” *in Proc. 28th Int. Conf. Software Engineering*, 2006, pp. 33–42.

[8] S. Bauer, “Das vernetzte Fahrzeug – Herausforderungen für die IT,” *Informatik Spektrum*, vol. 34, no. 1, pp. 38–41, Dec. 2010.

[9] A. Pretschner, M. Broy, I. H. Kruger, and T. Stauner, “Software Engineering for Automotive Systems: A Roadmap*,” in Proc. 2007 Future of Software Engineering*, pp. 55–71, May 2007.

[10] M. Glaab, W. Fuhrmann, and J. Wietzke, “Entscheidungskriterien für die Verteilung zukünftiger automotiver Anwendungen im Kontext vernetzter Fahrzeuge,” *in Proc. Mobilkommunikation 2011 - Technologien und Anwendungen - 16. ITG-Fachtagung*, Osnabrück, Germany, 2011, pp. 149–154.

[11] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, “M2M: From Mobile to Embedded Internet,” *IEEE Commun. Mag.*, vol. 49, no. 4, pp. 36–43, Apr. 2011.

[12] W. Huber, M. Lädke, and R. Ogger, “Extended floating-car data for the acquisition of traffic information,” *in Proc. 6th World Congr. Intelligent Transportation Systems*, pp.1-9, 1999.

[13] C. Bormann, A. P. Castellani, and Z. Shelby, “CoAP: An Application Protocol for Billions of Tiny Internet Nodes,” *IEEE Internet Comput.*, vol. 16, no. 2, pp. 62–67, 2012.

[14] D. C. Luckham, The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems, 5th ed. Boston, USA: Addison-Wesley Professional, 2002.

[15] R. Bruns, J. Dunkel, H. Masbruch, and S. Stipkovic, “Intelligent M2M: Complex event processing for machine-to-machine communication,” *Expert Systems with Applications*, vol. 42, no. 3, pp. 1235–1246, Feb. 2015.