Mobile Edge Computing Services for Dynamic Quality of Service Control

Evelina Pencheva, Ivaylo Atanasov
Technical University of Sofia
Sofia, Bulgaria
{enp, iia}@tu-sofia.bg

***Abstract*—Mobile Edge Computing (MEC) provides cloud computing capabilities in the radio access network. It enables applications that improve user's quality of experience and guarantee maximum utilization of radio network resources. Communication between MEC services and applications is based on Service-oriented Architecture (SOA). In this paper, capabilities for deployment of existing Parlay X Web Services in MEC environment are studied, and a new Web Service that provides access to User equipment-related radio network information is proposed.**

I. Introduction

Mobile Edge Computing (MEC) is a new technology that provides IT and cloud computing capabilities within the Radio Access Network [1]. It appears to be an important ingredient of 5G networks as it can improve user's quality of experience (QoE) and guarantee maximum utilization of radio network resources [2]. The processing logic is located close to base station (eNB) which reduces delays and allows timely reaction of dynamically changes in radio conditions. The vicinity to end users enables applications with high bandwidth and low latency requirements. For telecom operators MEC is a new source for revenue generation opportunities and it reduces the cost of data delivery while scaling their network [3], [4], [5]. MEC provides real-time network data such as radio conditions and network statistics for authorized applications to offer context-related services that can differentiate end user experience. A number of applications may be created and deployed at the mobile network edge like consumer-oriented services (gaming, augmented and assisted reality, cognitive assistance), operator and third party services (active device location tracking, big data, security, enterprise services), and network performance and QoE improvements (content cashing, performance optimization, video optimization, etc.) [6], [7]. A number of MEC use cases and deployment options are presented in [8]. A comprehensive survey of the state-of-the-art MEC research with a focus on joint radio-and-computational resource management is provided in [9].

ETSI, the European Telecommunications Standards Institute, defined MEC reference architecture, where MEC deployment can be inside the base station or at aggregation point within Radio Access Network (RAN) [10]. Minimal latency for many applications can be achieved by integrating MEC server inside the base station [11], [12].

MEC-service platform provides three types of middleware services: infrastructure services, radio network information services and traffic offload function [13]. Infrastructure services are used by applications for communications, service discovery and integration. Radio network information services provide authorized applications with low level real-time radio and network information related to users and cells. The traffic offload function prioritizes traffic and routes the selected, policy-based, user-data stream to and from applications that are authorized to receive the data.

Communication between MEC services and applications is based on Service-oriented Architecture (SOA). The applications access MEC services and other applications hosted on the MEC platform through Web Services (WS) Application Programming Interfaces (APIs). The aim is to reuse the existing APIs as much as possible.

The survey on MEC related works shows that the research is focused on MEC applications, but not on MEC services and the APIs that provide applications with radio resource management functionality. In this paper, we study the capabilities for deployment of Parlay X Application Driven Quality of Service (ADQ) WS in MEC environment [14]. We also propose a new WS that allows authorized MEC applications to manage User equipment (UE) context based on delivered radio network information. The functionality of Web Services is related to one UE and is based on S1AP functions [15]. The UE-associated services are associated with a UE-associated signaling connection maintained for the given UE.

The paper is structured as follows. In the next section use cases are presented where ADQ WS may be used and UE context information is required. In section III, a mapping of ADQ WS interfaces onto S1AP protocol is provided. Section IV presents a WS that allows authorized applications to receive notifications about UE context related events and to instruct eNB how to handle them. In section V, some WS implementation issues are discussed. The conclusion summarizes the authors' contribution and outlines the benefits of the proposed solution.

II. Use cases of dynamic QoS control

As to [13], traffic routing is a MEC platform essential functionality. This functionality allows authorized applications to inspect, modify, and shape selected uplink and/or downlink user plane traffic. Dynamic QoS control on end user connections may be implemented if the MEC platform supports features *UEIdentity* and *RadioNetworkInformation*. If the MEC platform supports *UEIdentity* feature it provides applications with functionality to register a token (representing a UE) or a list of tokens, and to set packet filters for routing traffic based on a token representing the UE. If the MEC platform supports

ISSN 2305-7254

RadioNetworkInformation feature, it provides appropriate up-to-date radio network information at the relevant granularity (e.g. per UE or per cell, per period of time).

The MEC applications, capable of dynamic QoS control, are generally aimed at improving performance of the network and user experience.

Examples of such application are orchestration of video streams and video optimization.

MEC can save backhaul capacity and provide high level performance and quality for video streams by locally produced video content and adding additional information. MEC platform collects data from local production devices and routes it to video orchestration application. Requests from UE to receive video are directed to the edge video orchestration application which routes the selected content to the user. MEC platform provides service for routing traffic form local production devices to the user.

MEC can improve user's QoE and guarantee maximum utilization of radio network resources by eliminating mobile content delivery inefficiencies. Video delivery is done via HTTP streaming, which is based on Transmission Control Protocol (TCP). TCP is not able to adapt to the changes of radio channel conditions, which leads to inefficient utilization of radio resources. An analytic application may derive radio network information available at the MEC platform and compute throughput guidance values [16]. Using the MEC routing services, the application may send the optimal bit rate to the video server to use given the radio conditions for a particular video stream or user.

Other examples include local content caching at the mobile edge, mobile backhaul optimization etc.

The dynamic QoS control may be provided by using APIs of Parlay X ADQ WS and access to UE related radio network information.

III. Application-driven QoS in MEC environment

The functionality for inspecting, modifying, and shaping selected uplink and/or downlink user plane traffic may be implemented by using Parlay X Application-Driven Quality of Service WS. ADQ is a service which enables applications to dynamically change the quality of service (e.g. bandwidth) available on end user network connections [14]. Changes in QoS may be applied on either a temporary basis (i.e. for a defined period of time), or as the default QoS to be applied for users each time they connect to the network. In the context of MEC routing service, the ApplicationQoS interface provides methods for dynamically controlling temporary QoS features in the network which will be active for a specified period of time, modifying an active temporary QoS Feature on an end user connection and self-care like operations. The invocation of interface operations results in initiation of S1AP procedures related to EUTRAN Radio Access Bearers (E-RABs).

The applyQoSFeature operation is used by Application to request a temporary QoS feature to be setup on the end user connection. Implemented in MEC platform this operation allows the Application to assign resources on Uu and S1 interfaces for one or several E-RABs and to setup corresponding Data Radio Bearers for a given UE. The operation initiates E-RAB Setup procedure, as shown in Fig. 1.

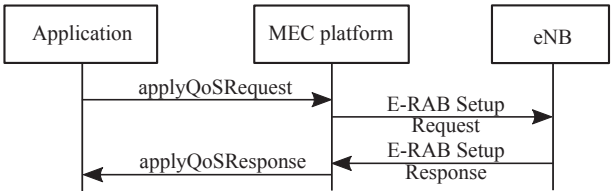

Fig. 1. The Application applies specific QoS to a UE connection

The Parlay X ADQ defines only DownStreamSpeedRate and UpStreamSpeedRate as QoS attributes. We suggest an extension of QoSFeatureProperties structure with elements indicating QoS parameters of bearers in Evolved Packet System, namely QoS Class identifier, Allocation and Retention Priority (ARP), Guaranteed Bit Rate (GBR) and Maximum Bit Rate (MBR) for GBR bearers and APN Aggregated Maximum Bit Rate (APN-AMBR) and UE-AMBR (TS 23.203).

```
<xs:schema  xmlns:xs="http://www.w3.org/2001/XMLSchema"
            xmlns="http://example.com"
            targetNamespace="http://example.com">
<xs:simpleType name="ActionValue">
 <xs:restriction base="xs:string">
  <xs:enumeration value="continue"/>
  <xs:enumeration value="changeQoS"/>
  <xs:enumeration value="reject"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="bitString">
 <xs:restriction base="xs:token">
  <xs:pattern value="[0-1]{0,}"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="TransportLayerAddress">
 <xs:restriction base="bitString">
  <xs:minLength value="1"/>
  <xs:maxLength value="160"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="GTP-TunnelEndPointID">
 <xs:restriction base="xs:hexBinary">
  <xs:minLength value="4"/>
  <xs:maxLength value="4"/>
 </xs:restriction>
</xs:simpleType>
<xs:simpleType name="ERABearerID">
 <xs:restriction base="xs:nonNegativeInteger">
  <xs:minInclusive value="1"/>
  <xs:maxInclusive value="15"/>
 </xs:restriction>
</xs:simpleType>
<xs:complexType name="ActionType">
 <xs:sequence>
  <xs:element name="ActionToPerform" type="ActionValue"/>
  <xs:element name="RoutingAddress" type="TransportLayerAddress"
              minOccurs="0" maxOccurs="1"/>
  <xs:element name="GTP-TEID" type="GTP-TunnelEndPointID"
              minOccurs="0" maxOccurs="1"/>
  <xs:element name="ERABID" type="ERABearerID"/>
 </xs:sequence>
</xs:complexType>
</xs:schema>
```

Fig. 2. XML scheme related to E-RAB QoS parameters

Fig. 2 shows the XML scheme related to E-RAB QoS parameters.

When the eNB reports unsuccessful establishment of an E-RAB, the cause value indicates the reason for unsuccessful establishment, e.g., “Radio resources not available”, “Failure in the Radio Interface Procedure.” The respective WS exception is “Insufficient connection resources.” If the eNB receives E-RAB Setup Request message containing incorrect E-RAB ID or QoS parameter values indicating QCI of GBR bearer, which does not contain GBR QoS information, then the eNB considers the establishment of the corresponding E-RAB as failed and the WS generates exception “Invalid input value.”

The modifyQoSFeature operation may be used by Application to alter the QoS attributes of an active temporary QoS feature instance. This operation initiates E-RAB modification procedure, as shown in Fig. 3.

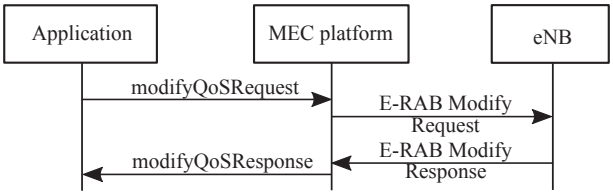

Fig. 3. The Application modifies the temporary QoS to UE connections

The service exception “Invalid input value” is raised when the eNB founds incorrect E-RAB ID.

The removeQoSfeature operation may be used by the Applications to release a temporary QoS Feature, which is currently active on the end user connection. This operation initiates E-RAB release procedure, as shown in Fig. 4.

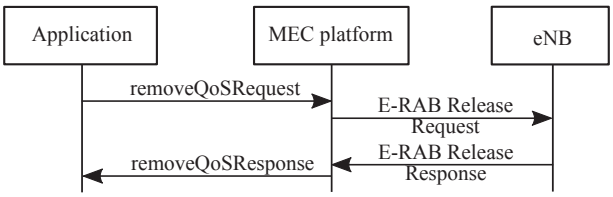

Fig. 4. The Application removes the temporary QoS to UE connections

The getQoSStatus operation falls in the category of a self-care and may be used to retrieve the status of an end user connection. The response to this method will contain information about the characteristics of the active E-RABs including information about the temporary QoS features that are currently active on the end user connection.

The getQoSHistory operation returns an historical list of all QoS transactions previously requested against an end user’s connection. The transactions to be returned may be filtered by specifying a QoS Feature identifier, a maximum number of transactions, a date/time limit, or additional filter criteria. The additional criteria may include E-RAB QoS information classified as to QCI, ARP, GBR, etc.

The ApplicationQoSNotificationManager interface is used by the Applications to manage their registration for notifications.

The startQoSNotification operation may be used by the Application to register its interest in receiving notifications of a specific event type(s) in context of specific end users. This operation starts monitoring and collecting radio network information, related to UE E-RABs. The stopQoSNotification operation is used by the Applications to stop receiving notifications by canceling an existing registration.

The ApplicationQoSNotification interface provides the operations for notifying the Application about the impact of certain events on QoS features that are active on the user connection when these events occurred. The notifyQoSEvent operation reports a network event that has occurred against user active QoS features. The QoSEvent type specifies the events that may occur on any active QoS features on the user connection(s). The “abnormal connection termination” event occurs when the dedicated E-RAB established by the Application are lost. The “normal connection termination” event occurs when the dedicated E-RAB established by the Application is released, or when all E-RABs are removed due to user inactivity (UE context release procedure is performed). The “temporary QoS feature released event” occurs when a temporary QoS feature that was active on an established E-RAB is released because the threshold set by one of the service attributes (e.g. elapsed duration) has been reached.

Fig. 5 shows an example of Application subscription and notification about QoS events.

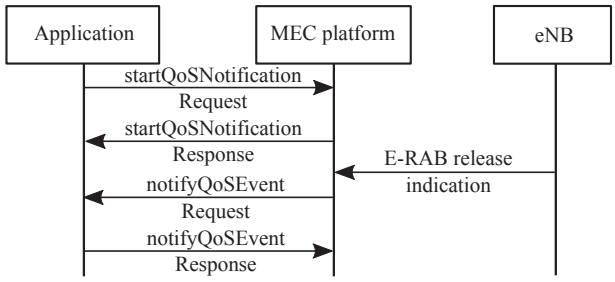

Fig. 5. Subscription for and notification about UE context modification

IV. UE CONTEXT NOTIFICATION WEB SERVICE

The functionality of UE Context Notification Web Service is based on the following functions of S1 AP:

- Initial Context Transfer function: This functionality is used to establish an S1UE context in the eNB, to setup the default IP connectivity, to setup one or more E-RAB(s) if requested by the MME, and to transfer Non Access Stratum signaling related information to the eNB if needed;
- UE Context Modification function: This functionality allows partly modification of the established UE Context;
- UE Context Resumption function: This functionality allows keeping the UE Context in the eNB for a UE in RRC_IDLE that has been enabled to use user plane optimization and to resume the radio connection without the need to re-establish the UE Context;
- S1 UE Context Release function: This functionality is responsible to manage the release of UE specific context in the eNB and the MME.

The UEContextManagement interface allows an authorized Application to receive notifications about UE context related events and to indicate how the context should be handled. The Application is notified in case of the following events:

- Initial context setup;
- eNB-initiated UE context release;
- MME-initiated UE context release;
- UE context modification;
- UE context modification indication;
- UE context suspend;
- UE context resume and release.

The invocation of handleContextSetup operation requests the Application to inform the MEC server how to handle the UE context on establishment of initial UE context. The Application may accept, change or reject the initial UE context setup ("continue," "changeQoS" or "reject" action), as shown in Fig. 6.

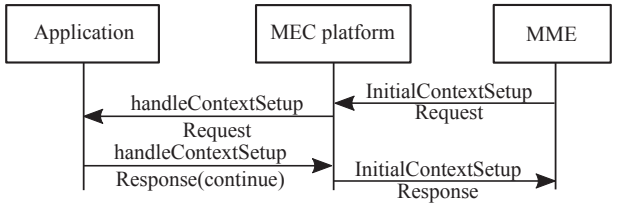

Fig. 6. The Application approves initial UE context establishment

The invocation of handleMMEContextRelease operation requests the Application to inform the MEC server how to handle the UE context when MME initiates a release of UE-associated logical S1-connection. The Application may accept or reject the UE context release.

The invocation of handleContextModified operation requests the Application to inform the MEC server how to handle the UE context when MME requests to partly modify the established UE Context. Application may accept, change or reject the UE context modification.

The invocation of handleContextResume operation requests the Application to inform the MEC server how to handle the UE context when eNB requests from MME to resume the UE-associated logical S1-connection. The Application may accept or reject the UE context resume.

The UEContextManagementManager interface enables applications to setup and tear down notification subscriptions for UE Contexts online. The startContextManagementNotification starts notifications to the application for given UE. The criteria parameter specifies the event specific criteria used by Application to define the event required. Only events that meet these criteria are notified. If the criteria parameter is not present, all UE Context events will be notified. The Application may end the UE Context notification using the stopContextManagementNotification operation.

UEContextNotification interface allows the Application to be notified about UE context related events. When UE Context events occur in the network, the Application may be notified of UE context related events. The Application does not have the ability to influence the UE Context, as UE Context management continues. Notifications are provided for Initial context setup, eNB-initiated UE context release, MME-initiated UE context release, UE context modification, UE context modification indication, UE context suspend, UE context resume and release.

The notifyContextSetup operation informs the Application that the necessary overall initial UE context including E-RAB context, the security key, handover restriction list, UE radio capability and UE security capabilities etc. are established. The notifyENBContextRelease operation informs the Application that eNB requests the MME to release the UE-associated logical S1 connection due to E-UTRAN generated reasons. The notifyMMEContextRelease operation informs the Application that MME orders the release of the UE-associated logical S1 connection due to various reasons, e.g. completion of a UE transaction. The notifyContextModified operation informs the Application that MME requests partly to modify the established UE Context. The notifyContextModifiedInd operation informs the Application that eNB requests the modifications on the established UE Context. The notifyContextSuspend operation informs the Application that eNB requests to suspend the UE context, the UE-associated logical S1-connection and the related bearer contexts in the E-UTRAN. The notifyContextResume operation informs the Application that eNB indicates to the MME that the UE has resumed the suspended RRC connection and to request the MME to resume the UE context, UE-associated logical S1-connection and the related bearer contexts.

The UEContextNotificationManager interface enables applications to set up and tear down notifications for UE contexts online. The startContextNotification operation starts notifications to the Application for given UE. The Application may end the UE context notification using the stopContextNotification operation.

Fig. 7 illustrates subscription for and notification about UE context modifications.

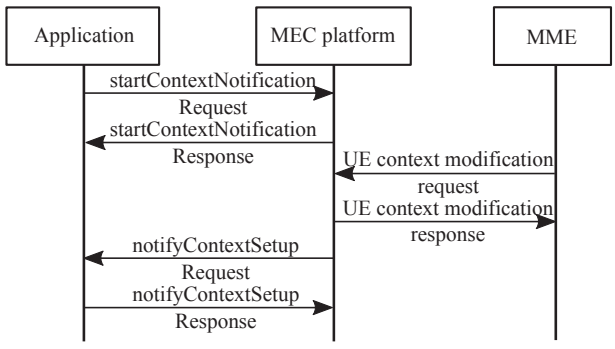

Fig. 7. Subscription for and notification about UE context modification

V. Implementation Issues

The MEC platform needs to translate Web Service messages into S1AP protocol messages and vice versa. In addition, it has to maintain synchronized the UE context state models

as seen by the network and by the Application. Fig. 8 shows the UE context state model as seen by the Application.

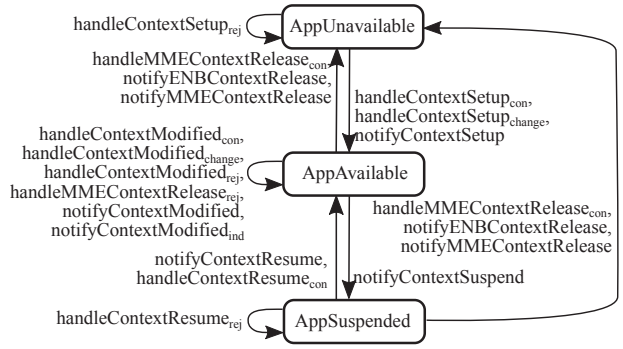

Fig. 8. UE context state model as seen by Application

In AppUnavailable state, the UE context is not available at the eNB. Upon notification of initial UE context setup, the Application may accept the establishment of the S1AP connection without any changes, or may change some E-RAB related QoS parameters, or may reject the UE context setup. In AppAvailable state, the UE context is available at the eNB and the Application may be notified about UE context modification, suspend or release request and indicate how to handle the request. In AppSuspended state, the UE context is suspended, and the Application may be notified about UE context resume or release request, and instructs the MEC platform about the necessary action.

We use the mathematical formalism of Labeled Transition Systems (LTSs) to describe the UE context state models.

An LTS is defined as a quadruple of set of states, set of inputs, set of transitions, and an initial state.

By $C_{App} = (S_{App}, Inp_{App}, \rightarrow_{App}, s^0_{App})$ it is denoted an LTS representing the Application's view on device state where:

$$S_{App}=\{AppUnavailable\left[s_1^A\right], AppAvailable\left[s_2^A\right], AppSuspended\left[s_3^A\right]\};$$

$$Inp_{App}=\{handleContextSetup_{con}\left[t_1^A\right], handleContextSetup_{change}\left[t_2^A\right], handleContextSetup_{rej}\left[t_3^A\right], notifyContextSetup\left[t_4^A\right], handleMMEContextRelease_{con}\left[t_5^A\right], handleMMEContextRelease_{rej}\left[t_6^A\right], handleContextModified_{con}\left[t_7^A\right], handleContextModified_{change}\left[t_8^A\right], handleContextModified_{rej}\left[t_9^A\right], notifyContextModified_{ind}\left[t_{10}^A\right], notifyContextModified\left[t_{11}^A\right], notifyContextSuspend\left[t_{12}^A\right], notifyContextResume\left[t_{13}^A\right], notifyContextResume_{con}\left[t_{14}^A\right], notifyContextResume_{rej}\left[t_{15}^A\right], notifyENBContextRelease\left[t_{16}^A\right], notifyMMEContextRelease\left[t_{17}^A\right]\};$$

$$\rightarrow_{App}=\{\left(s_1^A t_3^A s_1^A\right), \left(s_1^A t_1^A s_2^A\right), \left(s_1^A t_2^A s_2^A\right), \left(s_1^A t_4^A s_2^A\right), \left(s_2^A t_5^A s_1^A\right), \left(s_2^A t_{16}^A s_1^A\right), \left(s_2^A t_{17}^A s_1^A\right), \left(s_2^A t_7^A s_2^A\right), \left(s_2^A t_8^A s_2^A\right), \left(s_2^A t_9^A s_2^A\right), \left(s_2^A t_{10}^A s_2^A\right), \left(s_2^A t_{11}^A s_2^A\right), \left(s_2^A t_{12}^A s_3^A\right), \left(s_3^A t_{13}^A s_2^A\right), \left(s_3^A t_{14}^A s_2^A\right), \left(s_3^A t_{15}^A s_3^A\right), \left(s_3^A t_5^A s_1^A\right), \left(s_3^A t_{16}^A s_1^A\right), \left(s_3^A t_{17}^A s_1^A\right)\};$$

$$s^0_{App}=\{s_1^A\}.$$

Short notation of state names and input names are given in brackets. Fig. 9 shows the UE context state model as seen by the eNB. The model is in compliance with S1AP procedures.

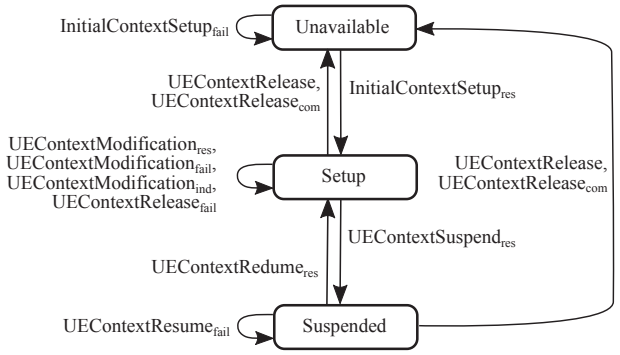

Fig. 9. UE context state model as seen by eNB

By $C_{eNB} = (S_{eNB}, Inp_{eNB}, \rightarrow_{eNB}, s^0_{eNB})$ it is denoted an LTS representing the eNB's view on device state where:

$$S_{eNB}=\{Unavailable\left[s_1^N\right], Available\left[s_2^N\right], Suspended\left[s_3^N\right]\};$$

$$Inp_{eNB}=\{InitialContextSetup_{fail}\left[t_1^N\right], InitialContextSetup_{res}\left[t_2^N\right], UEContextRelease\left[t_3^N\right], UEContextRelease_{com}\left[t_4^N\right], UEContextModification_{res}\left[t_5^N\right], UEContextModification_{fail}\left[t_6^N\right], UEContextModification_{ind}\left[t_7^N\right], UEContextRelease_{fail}\left[t_8^N\right], UEContextSuspend_{res}\left[t_9^N\right], UEContextResume_{res}\left[t_{10}^N\right], UEContextResume_{fail}\left[t_{11}^N\right]\};$$

$$\rightarrow_{eNB}=\{(s_1^N t_1^N s_1^N), (s_1^N t_2^N s_2^N), (s_2^N t_3^N s_1^N), (s_2^N t_4^N s_1^N), (s_2^N t_5^N s_2^N), (s_2^N t_6^N s_2^N), (s_2^N t_7^N s_2^N), (s_2^N t_8^N s_2^N), (s_2^N t_9^N s_3^N), (s_3^N t_{10}^N s_2^N), (s_3^N t_{11}^N s_3^N), (s_3^N t_3^N s_1^N), (s_3^N t_4^N s_1^N)\}$$

$$s^0_{eNB}=\{s_1^N\}.$$

We use the concept of weak bisimulation to formally verify the suggested models.

Proposition 1: The labeled transition systems C_{App} and C_{eNB} are weakly bisimilar.

Proof: As to definition of weak bisimulation, provided in [16], it is necessary to identify a bisimilar relation between the states of both LTSs and to identify respective matching between transitions.

Let U_{AppeNB} denote a relation between S_{App} and S_{eNB}, where $U_{AppeNB} = \{(s_1^A, s_1^N), (s_2^A, s_2^N), (s_3^A, s_3^N)\}$. Then for

the following network events we identify the respective transitions between states of C_{App} and C_{eNB}:

1) In case of unsuccessful initial UE context setup, for $(s_1^A t_3^A s_1^A) \exists (s_1^N t_1^N s_1^N)$;
2) In case of successful initial UE context setup, for $(s_1^A t_1^A s_2^A) \exists (s_1^N t_2^N s_2^N)$, for $(s_1^A t_2^A s_2^A) \exists (s_1^N t_2^N s_2^N)$ and for $(s_1^A t_4^A s_2^A) \exists (s_1^N t_2^N s_2^N)$;
3) In case of MME initiated UE context release, for $(s_2^A t_5^A s_1^A) \exists (s_2^N t_4^N s_1^N)$ and $(s_2^A t_{17}^A s_1^A) \exists (s_2^N t_4^N s_1^N)$;
4) In case of eNB initiated UE context release, for $(s_2^A t_{16}^A s_1^A) \exists (s_2^N t_3^N s_1^N)$;
5) In case of successful UE context modification, that is initiated by MME, for $(s_2^A t_7^A s_2^A) \exists (s_2^N t_5^N s_2^N)$, and for $(s_2^A t_8^A s_2^A) \exists (s_2^N t_5^N s_2^N)$;
6) In case of successful UE context modification, that is initiated by eNB, for $(s_2^A t_{11}^A s_2^A) \exists (s_2^N t_7^N s_2^N)$;
7) In case of unsuccessful UE context modification, for $(s_2^A t_9^A s_2^A) \exists (s_2^N t_6^N s_2^N)$;
8) In case of unsuccessful UE context release, for $(s_2^A t_6^A s_2^A) \exists (s_2^N t_8^N s_2^N)$;
9) In case of UE context suspend, for $(s_2^A t_{12}^A s_3^A) \exists (s_2^N t_9^N s_3^N)$;
10) When the UE context is resumed successfully, for $(s_3^A t_{13}^A s_2^A) \exists (s_3^N t_{10}^N s_2^N)$ and $(s_3^A t_{14}^A s_2^A) \exists (s_3^N t_{10}^N s_2^N)$;
11) In case of unsuccessful UE context resume, for $(s_3^A t_{15}^A s_3^A) \exists (s_3^N t_{11}^N s_3^N)$;
12) In case of eNB-initiated release of suspended UE context, for $(s_3^A t_3^A s_1^A) \exists (s_3^N t_3^N s_1^N)$;
13) In case of MME-initiated release when UE context is suspended, for $(s_3^A t_5^A s_1^A) \exists (s_3^N t_4^N s_1^N)$ and for $(s_3^A t_{17}^A s_1^A) \exists (s_3^N t_4^N s_1^N)$;
14) In case of eNB-initiated UE context modification, for $(s_2^A t_{10}^A s_2^A) \exists (s_2^N t_7^N s_2^N)$.

Therefore C_{App} and C_{eNB} are weakly bisimilar. ■

VI. Conclusion

MEC technology distributes cloud intelligence close to the mobile network edge. It enables delay sensitive and context aware applications which optimize network resource utilization and improves end user experiences.

This paper presents a study of capabilities for deployment of existing Parlay X Application-driven QoS Web Service in MEC environment. The functionality of ADQ API is mapped onto S1AP protocol functions for E-RAB management and context management. Further, we propose a new WS which provides access to radio network information and allows management of UE context. WS implementation issues are considered. We present a method for formal verification of MEC platform which needs to maintain in synchronized manner state machines representing both the Application and S1AP protocol view on UE related information.

Reuse of existing API allows application developers and content providers to flexibly create and deploy innovative applications and services. The access of up-to-date radio network information requires development of new APIs which can alleviate network congestions and can efficiently serve local purposes.

Acknowledgement

The research is conducted under the grant of project DH07/10-2016, funded by National Science Fund, Ministry of Education and Science, Bulgaria.

References

[1] P. Corcoran and S.K. Datta, “Mobile-Edge Computing and the Internet of Things for Consumers: Extending cloud computing and services to the edge of the network”, *IEEE Consumer Electronics Magazine*, vol.5, no.4, Oct.2016, pp. 73–74.

[2] T. Tran, A. Hajisami, P. Pandey and D. Pompili, “Collaborative Mobile Edge Computing in 5G Networks: New Paradigms, Scenarios, and Challenges”, *Cornell University Library*, arXiv:1612.03184, 2016, pp. 1–9.

[3] Y. Chen and L. Ruchenbusch, “Mobile Edge Computing: Brings the Value Back to Networks”, *IEEE Software Defined Networks Newsletter*, Mar.2016, Web: http://sdn.ieee.org/newsletter/march-2016/mobile-edge-computing-bring-the-values-back-to-networks

[4] R. Roman, J. Lopez and M. Mambo, “Mobile edge computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges”, *Future Generation Computer Systems*, in press.

[5] M.T. Beck, S. Feld, C. Linnhhoff-Popien and U. Pützschler, “Mobile Edge Computing”, *Informatik-Spektrum*, vol.39 (2), 2016, pp. 108–114.

[6] T. Tran, P. Pandey, A. Hajisami and D. Pompili, “Collaborative Multi-bitrate Video Caching and Processing in Mobile-Edge Computing Networks”, *Cornell University Library*, arXiv:1612.01436, 2016, pp. 1–8.

[7] A. Ahmed and E. Ahmed, “A Survey on Mobile Edge Computing”, *10th IEEE Int. Conf. on Intelligent Systems and Control*, 2016, pp. 1–8.

[8] G. Brown, “Mobile Edge Computing Use Cases and Deployment Options”, *Juniper White Paper*, 2016, pp. 1–10, Web: www.juniper.net/assets/us/en/local/pdf/whitepapers/2000642-en.pdf

[9] Y. Mao, C. You, J. Zhang, K. Huang and K. Letaief, “Mobile Edge Computing: Survey and Research Outlook”, *IEEE Communications Surveys & Tutorials*, arXiv:1701.01090 [cs.IT], 2017, pp. 1–30.

[10] ETSI GS MEC 003, “Mobile Edge Computing (MEC); Framework and Reference Architecture”, v1.1.1, 2016

[11] D. Sarria, D. Park and M. Jo, “Recovery for overloaded mobile edge computing”, *Future Generation Computer Systems*, vol.70, 2017, pp. 138–147.

[12] M. Beck, M. Werner, S. Feld and T. Schimper, “Mobile Edge Computing: A Taxonomy”, *6-th Int. Conf. on Advances in Future Internet*, 2014, pp. 48–54.

[13] ETSI GS MEC 002, “Mobile Edge Computing; Technical Requirements”, v1.1.1, 2016.

[14] 3GPP TS 29.199-17, “Open Service Access (OSA); Parlay X Web Services; Part 17: Application-driven Quality of Service”, Rel.9, v9.0.0, 2009.

[15] 3GPP TS 36.300, “Evolved Universal Terrestrial Radio Access (EUTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2”, Rel.14, v14.0.0, 2016.

[16] A. Jain, A. Terzis, N. Sprecher, P. Szilagyi and H. Flinck, “Mobile Throughput Guidance Inband Signaling Protocol”, *Internet draft, IETF*, 2015

[17] A. Chebieb and Y. A. Ameur, “Formal Verification of Plastic User Interfaces Exploiting Domain Ontologies,” Int. Symp. on Theoretical Aspects of Software Engineering, Nanjing, 2015, pp. 79–86.

[18] D. Escrig, J. Keiren and T. Willemse, “Games for Bisimulations and Abstraction”, *Cornell University Library*, arXiv:1611.00401