MOBILE EDGE COMPUTING APPLICATIONS FOR BANDWIDTH MANAGEMENT[1]

Evelina Pencheva

Technical University of Sofia
e-mail: enp@tu-sofia.bg
Bulgaria

Abstract: With the ubiquitous penetration of Internet of Things new applications are expected to benefit from bringing the cloud close to the network edge. Mobile Edge Computing (MEC) allows monitoring radio access network and provides critical information related to users and cells to analytic real-time applications. Deployment of MEC applications may be facilitated by existence of horizontal service platforms enabling MEC application orchestration. In this paper, the MEC service for bandwidth management is studied. Generic functionality is explored and semantic data annotation is synthesized. An approach to describe applications based on semantic annotation and a method for automatic detection and resolution of undesired interaction between MEC applications are proposed.

Key words: Quality of Service; online charging; behavioural models; semantic data; service orchestration; reasoning

1. INTRODUCTION

With the ubiquitous penetration of Internet of Things (IoT) the necessity of applications with requirements for real-time operation, low latency requirements, and high quality of service increases. Mobile Edge Computing (MEC) is an IT technology that brings cloud computing capabilities close to the Radio Access Network [1]. MEC contributes to reducing latency, improvement of network operation and end user experience. Potential MEC-based applications include smart mobility, connected vehicles, emergency response, smart cities, content distribution, and location services [2, 3, 4].

MEC applications are instantiated on the virtualization infrastructure of the mobile edge server which is collocated with the radio base station at aggregation

[1] The research is conducted under the grant of project ДН07/10-2016, funded by National Science Fund, Ministry of Education and Science, Bulgaria.

points [5]. The mobile edge orchestrator is the core functionality of MEC management. In addition to functions related to management of the MEC application life cycle, it needs to orchestrate MEC services, including detecting and resolving undesired interactions between MEC applications.

Undesired interaction manifests itself as a function of application logics which is neither exactly the sum of every application logic nor behaves as expected. Despite of the progress in developing approaches for modelling, detecting, and resolving such kind of interactions, there is a lack of sufficient knowledge on the kind of interactions that occur in real-world IoT [6, 7, 8].

In this paper, MEC applications that may be used for bandwidth management are studied. Application logic is mapped onto network functionality for quality of service (QoS) control. In order to separate the application-specific functionality from the generic bandwidth management functions, semantic annotation on bandwidth management data is defined. A method for MEC application orchestration is proposed. The approach is based on author's work [9, 10].

The paper is structured as follows. Section II presents the MEC service for bandwidth management and its deployment in the network. Section III describes MEC applications and considers some issues related to mapping of application logic onto respective network functionality. Semantic annotation of bandwidth management data is formally described in Section IV. Possible undesired interactions between MEC applications and their resolution are discussed in Section V. The conclusion outlines implementation aspects of the proposed method for service orchestration and highlights its benefits.

2. MEC SERVICE FOR BANDWIDTH MANAGEMENT

MEC platform is a collection of essential functionality required to run MEC applications on a particular virtualization infrastructure and to enable them to provide and consume MEC services. The standardization of MEC is ongoing. The MEC platform should allow authorized MEC applications to provide services that can be consumed by the platform or by authorized MEC applications running on the MEC host. It implements Service Capability Exposure Function (SCEF), which is the key entity for service capability exposure that provides a means to securely expose the services and capabilities via Application Programming Interfaces (APIs) [11].

MEC platform may support feature called *BandwidthManager* [12]. When the *BandwidthManager* feature is supported, MEC platform, or a dedicated MEC application, enables authorized MEC applications to register statically and/or dynamically its bandwidth requirements and/or priority, and may allocate bandwidth and/or assign priority to any session or to any application. This feature may be implemented in conjunction with Policy and Charging Control (PCC) functionality.

PCC provides a mechanism to authorize and control the usage of the bearer traffic and it is used for ensuring coherent charging [13]. The Policy and Charging Rule Function (PCRF) encompasses policy control decision and flow based charging control functions. In case of online charging, the MEC platform needs to interact with Online Charging System (OCS) for charging information and instructions.

Currently no bandwidth management interfaces are defined, but the applications that may benefit from this MEC middleware service are mainly aimed at improving network performance and user's quality of experience. Examples of applications include mobile video delivery optimization using throughput guidance for TCP, mobile backhaul optimization based on coordination between backhaul network and radio access network, video orchestration where the content is produced as close to end user and combined from multiple sources, etc. [14, 15, 16].

Fig.1 shows the architecture for deployment of MEC applications for bandwidth management.

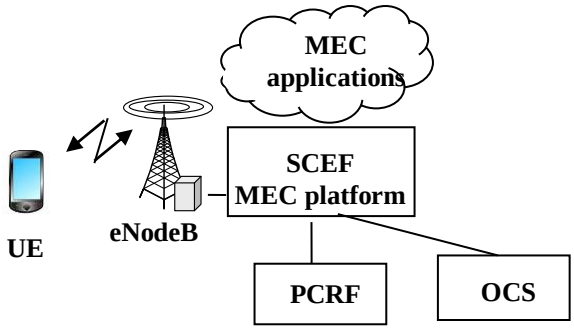

Fig. 1. Architecture for deployment of MEC applications for PCC

In the next section, bandwidth applications that use real-time radio network information related to UE are presented. The applications improve the connectivity management parameter available on UE connections based on policies.

3. GENERIC FUNCTIONALITY FOR BANDWIDTH MANAGEMENT

Let us consider a bandwidth manager (BWM) application which uses traps for diagnosis and monitoring the UE connectivity. When the UE communicates over the wireless network, it is important to collect the measurements data (e.g. signal quality information) for network optimization. Open Mobile Alliance (OMA) has defined a number of diagnostic and monitoring traps for this purpose [17]. The Received power trap can help the network optimization process by triggering the application logic when the received power of the UE drops below an application-specific value (TrapActivePower). Whenever the UE's received power rises above another application-specific value (TrapInactivePower), the application is notified. Further,

the BWM application may control the UE bearer usage based on the available balance of the subscriber (UE owner). The application defines two balance thresholds. If the subscriber's balance is under the first threshold, then it is considered to be low. In case the subscriber's balance is low, the application requires usage of "cheaper" bearer. When the low balance decreases under the second threshold, the application disconnects the UE. Fig. 2 shows the BWM application logic.

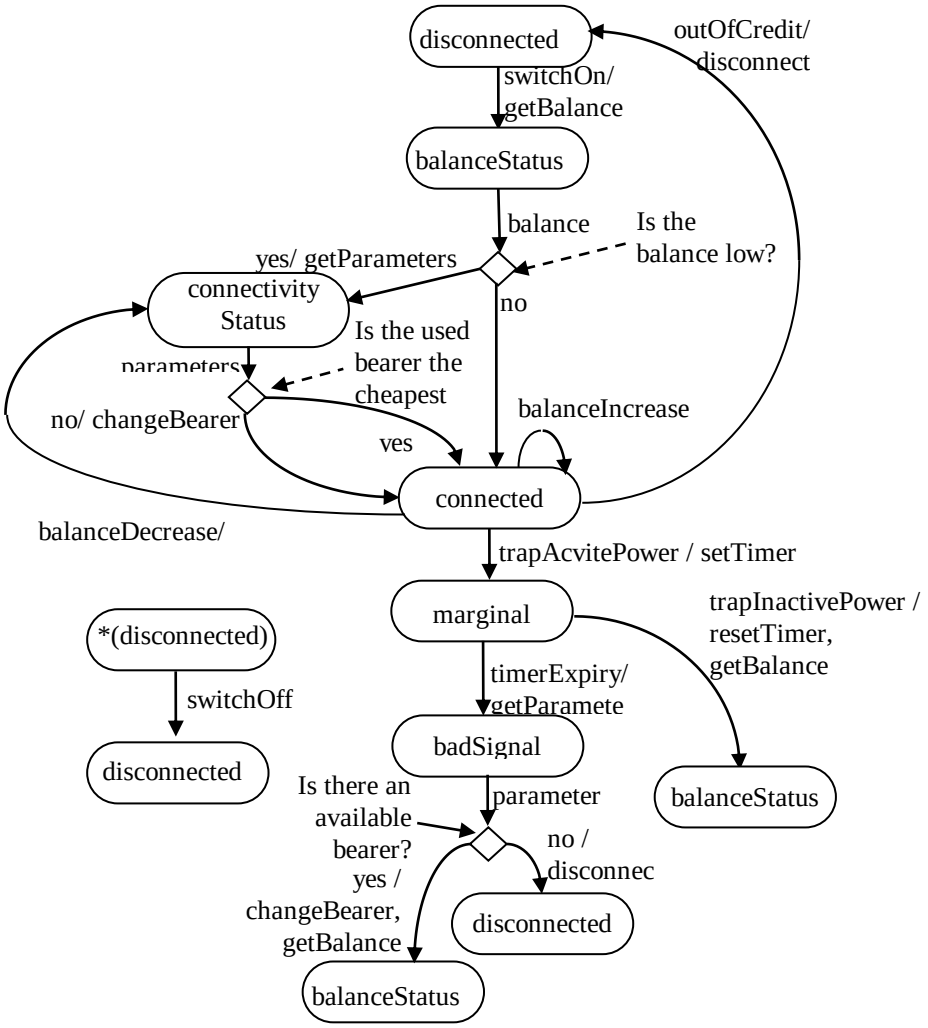

Fig. 2. Balance-based bandwidth management state model

In *disconnected* state, the UE is not connected to the network. When the UE is switched on, it connects to the network. In *balanceStatus* state, the application waits for the subscriber's balance status. If the balance is low, then the application queries

the UE about its connectivity parameters. In *connectivityStatus* state, the application waits for UE connectivity parameters. In *connectivityStatus* state, if the UE does not use the cheapest bearer, then the application request the UE to change the bearer. In *connected* state, the balance may increase or decrease. If the subscriber is out of credit, the application disconnects the UE. When the signal strength of the used bearer drops, the Receive power trap becomes active and the state becomes *marginal*. In *marginal* state, if the signal strength rises, the Receive power trap becomes inactive and the state becomes *connected*. If the *marginal* state is stable, then the state becomes *badSignal*. In *badSignal* state, the application queries the UE about its connectivity parameters, and if there is an available bearer, then the application requests the UE to change the used bearer. In *badSignal* state, if there is no available bearer, the application disconnects the UE. In any state but *disconnected*, the UE may be switched off.

In order to implement the BWM application logic, the network applies usage monitoring for the accumulated usage of network resources for the UE. This capability is required for enforcing dynamic policy decisions based on the resource usage in real-time. The network enforces policies based on subscriber spending limits [13]. The OCS maintains policy counter(s) to track spending for a subscription.

Fig.3 shows the network view on the resources used by the UE.

In *idle* state, the UE is not involved in a session. When the UE establishes a session, the PCRF starts monitoring the resource usage. The OCS makes information regarding the UE owner's spending available to the PCRF using spending limit reports. The usage monitoring thresholds shall be based either on time, or on volume. In *inSpendingLimits* state, the usage monitoring thresholds are not reached, the session may be modified. When a notification is received that usage monitoring thresholds are reached, the state becomes *outOfSpendingLimits*. Following a predefined policies, the network may terminate the session. The session may be terminated by the application also.

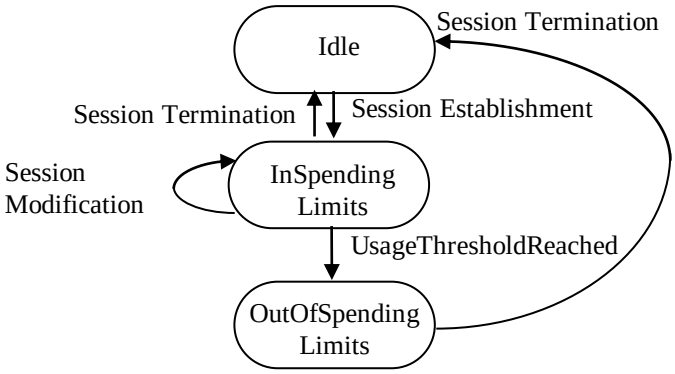

Fig. 3. Model representing the network view on the UE resource usage

Both models representing the application and network view on the sate of resources used by the UE need to be synchronized. Both models are formally described using the mathematical formalism of Labelled Transition Systems (LTS).

By BWM_{App}= (S_{App}, Inp_{App}, $\rightarrow_{App}$, s^0_{App}) it is denoted an LTS representing the BWM application view on bandwidth management state where:

S_{App} = { disconnected[s_1^A], balanceStatus[s_2^A], connectivityStatus[s_3^A], connected [s_4^A], marginal[s_5^A], badSignal[s_6^A],};

Inp_{App} = { switchOn[t_1^A], balance$_{low}$ [t_2^A], parameters$_{UsedCheap}$[t_3^A], parameters$_{NotCheap}$[t_4^A], balanceIncrease[t_5^A], trapActivePower[t_6^A], balanceDecrease[t_7^A], outOfCredit[t_8^A], trapInactivePower[t_9^A], timerExpiry[t_{10}^A], parameters$_{HasAvailable}$[t_{11}^A], parameters$_{NoAvailable}$[t_{12}^A], switchOff[t_{13}^A], balance$_{high}$[t_{14}^A]};

$\rightarrow_{App}$ = { (s_1^A t_1^A s_2^A), (s_2^A t_2^A s_3^A), (s_3^A t_3^A s_4^A), (s_3^A t_4^A s_4^A), (s_2^A t_{14}^A s_4^A), (s_4^A t_7^A s_3^A), (s_4^A t_5^A s_4^A), (s_4^A t_8^A s_1^A), (s_4^A t_6^A s_5^A), (s_5^A t_{10}^A s_6^A), (s_5^A t_9^A s_2^A), (s_6^A t_{11}^A s_2^A), (s_6^A t_{12}^A s_1^A), (s_2^A t_{13}^A s_1^A), (s_3^A t_{13}^A s_1^A), (s_4^A t_{13}^A s_1^A), (s_5^A t_{13}^A s_1^A), (s_6^A t_{13}^A s_1^A),}

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

Short notations of state and transition names are given in brackets.

By PCC_N= (S_N, Inp_N, $\rightarrow_N$, s^0_N) it is denoted an LTS representing the network view on resource state used by the UE, where:

S_N = { idle[s_1^N], inSpendingLimits[s_2^N], outOfSpendingLimits[s_3^N]};

Inp_N = { sessionEstablishment[t_1^N], sessionTermination[t_2^N], sessionModification[t_3^N] usageThresholdReached[t_4^N]};

$\rightarrow_N$ = { (s_1^N t_1^N s_2^N), (s_2^N t_2^N s_1^N), (s_2^N t_3^N s_2^N), (s_2^N t_4^N s_3^N), (s_3^N t_2^N s_1^N)}

s^0_N = { s_1^N }.

The concept of weak bisimulation is used to formally verify the suggested models [18].

Proposition: The labelled transition systems BWM_{App} and PCC_N are weakly bisimilar.

Proof: As to definition of weak bisimulation, provided in [18], it is necessary to identify a bisimilar relation between the states of both LTSs and to identify

respective matching between transitions. Let U_{AppN} is a relation between the states of BWM_{App} and PCC_{N} and U_{AppN}= {(disconnected, idle), (connected, inSpendingLimits)}, then:

1. In case UE connects to the network, the balance is low and UE uses the cheapest bearer, for $(s_1^A\ t_1^A\ s_2^A)$, $(s_2^A\ t_2^A\ s_3^A)$, and $(s_3^A\ t_3^A\ s_4^A) \exists (s_1^N\ t_1^N\ s_2^N)$.

2. In case UE connects to the network, the balance is low, and UE changes the used bearer with the cheaper one, for $(s_1^A\ t_1^A\ s_2^A)$, $(s_2^A\ t_2^A\ s_3^A)$, and $(s_3^A\ t_4^A\ s_4^A) \exists (s_1^N\ t_1^N\ s_2^N)$.

3. In case UE connects to the network and the balance is not low, for $(s_1^A\ t_1^A\ s_2^A)$ and $(s_2^A\ t_{14}^A\ s_4^A) \exists (s_1^N\ t_1^N\ s_2^N)$.

4. In case the subscriber is out of credit, for $(s_4^A\ t_8^A\ s_1^A) \exists (s_2^N\ t_4^N\ s_3^N)$ and $(s_3^N\ t_2^N\ s_1^N)$.

5. In case of stable low signal level when the UE changes the used bearer, for $(s_4^A\ t_6^A\ s_5^A)$, $(s_5^A\ t_{10}^A\ s_6^A)$, $(s_6^A\ t_{11}^A\ s_2^A)$, $(s_2^A\ t_2^A\ s_3^A)$, and $(s_3^A\ t_3^A\ s_4^A) \exists (s_2^N\ t_3^N\ s_2^N)$; or $(s_4^A\ t_6^A\ s_5^A)$, $(s_5^A\ t_{10}^A\ s_6^A)$, $(s_6^A\ t_{11}^A\ s_2^A)$, $(s_2^A\ t_2^A\ s_3^A)$, and $(s_3^A\ t_4^A\ s_4^A) \exists (s_2^N\ t_3^N\ s_2^N)$.

6. In case of temporary low signal level when the UE changes the used bearer, for $(s_4^A\ t_6^A\ s_5^A)$, $(s_5^A\ t_9^A\ s_2^A)$, $(s_2^A\ t_2^A\ s_3^A)$, and $(s_3^A\ t_3^A\ s_4^A) \exists (s_2^N\ t_3^N\ s_2^N)$; or $(s_4^A\ t_6^A\ s_5^A)$, $(s_5^A\ t_9^A\ s_2^A)$, $(s_2^A\ t_2^A\ s_3^A)$, and $(s_3^A\ t_4^A\ s_4^A) \exists (s_2^N\ t_3^N\ s_2^N)$.

7. In case of stable low signal level when there is no available bearer, for $(s_4^A\ t_6^A\ s_5^A)$,$(s_5^A\ t_{10}^A\ s_6^A)$ and $(s_6^A\ t_{12}^A\ s_1^A) \exists (s_2^N\ t_2^N\ s_1^N)$.

8. In case the UE is switched off, for $(s_2^A\ t_{13}^A\ s_1^A) \exists (s_2^N\ t_2^N\ s_1^N)$; or $(s_3^A\ t_{13}^A\ s_1^A) \exists (s_2^N\ t_2^N\ s_1^N)$; or $(s_4^A\ t_{13}^A\ s_1^A) \exists (s_2^N\ t_2^N\ s_1^N)$; or $(s_4^A\ t_{13}^A\ s_1^A) \exists (s_2^N\ t_2^N\ s_1^N)$; or $(s_5^A\ t_{13}^A\ s_1^A) \exists (s_2^N\ t_2^N\ s_1^N)$; or $(s_6^A\ t_{13}^A\ s_1^A) \exists (s_2^N\ t_2^N\ s_1^N)$.

10. In case of balance increase, for $(s_4^A\ t_5^A\ s_4^A) \exists (s_2^N\ t_3^N\ s_2^N)$.

11. In case of balance decrease, for $(s_4^A\ t_7^A\ s_3^A)$ and $(s_3^A\ t_3^A\ s_4^A) \exists (s_2^N\ t_3^N\ s_2^N)$; or for $(s_4^A\ t_7^A\ s_3^A)$ and $(s_3^A\ t_4^A\ s_4^A) \exists (s_2^N\ t_3^N\ s_2^N)$.

Therefore BWM_{App} and PCC_{N} are weakly bisimilar.■

OMA defined diagnosis and monitoring traps that may be used for bandwidth management include Geo trap, Data speed trap, QoS trap and Call drop trap [17]. Geo trap may trigger the application logic when the UE enters or exits a specific area, QoS trap is used to monitor the received QoS, Data speed trap monitors the data speeds experienced by the UE in uplink and downlink, and the Call drop trap may trigger the application logic when a call drop occurs in the predefined period.

Different applications for bandwidth management may be created based on OMA diagnosis and monitoring traps.

4. SEMANTIC ANNOTATION FOR BANDWIDTH MANAGEMENT

In order to provide basic service capability for bandwidth management it is important to separate the application-specific functionality from the generic bandwidth management functions. It may be achieved by provisioning of semantic annotation on bandwidth management data. The support of semantic annotation enables re-use of data by many applications and simplifies the application configuration.

Description logic is used to define basic concepts and their roles in providing bandwidth management functionality.

The approach to definition of atomic concepts is to represent the UE states and bearer related facts in the bandwidth management models as concepts.

Let us assume that there is a finite set of bearer indices which represent the possible bearers that may be used by a particular UE. The following concepts related to basic bandwidth management may be defined:

- *disconnected*, UE is disconnected;
- $connected_b$, UE is connected using bear *b*;
- $marginal_b$, UE's received power of used bearer *b* is below an application-specific value;
- $badSignal_b$, UE needs to change the used bearer *b*;
- $available_b$, bear *b* is available;
- *unavailable*, there are no available bearers.

The transitions that change the UE state are defined as atomic roles:

- *switchOn*, the UE is switched on;
- *switchedOff*, the UE is switched off;
- *signalDrop*, received power of used bearer drops below application-specific value;
- *signalRise*, received power of used bearer rises above application-specific value;
- $changeBearer_b$, the application requests the UE to change the bearer with *b*;
- *disconnect*, the application disconnects the UE from the network;
- *timerExpiry*, time guarded hysteresis of the received power is over.

The terminology box contains expressions showing the changes in BWM model and statements specifying the relationship between the events that cause transitions.

$$disconnected \sqsubseteq \exists switchOn.connected_b \quad (1)$$

$$connected_b \sqsubseteq \exists getParameters.connectivityStatus_b \quad (2)$$

$$connectivityStatus_b \sqsubseteq \exists parameters.(connected_b \sqcap available_c) \quad (3)$$

$$connectivityStatus_b \sqsubseteq \exists parameters.(connected_b \sqcap unavailable) \quad (4)$$
$$connected_b \sqsubseteq \exists signalDrop.marginal_b \quad (5)$$
$$marginal_b \sqsubseteq \exists signalRise.connected_b \quad (6)$$
$$marginal_b \sqsubseteq \exists timerExpiry.badSignal_b \quad (7)$$
$$badSignal_b \sqsubseteq \exists getParameters.badSignal_b \quad (8)$$
$$badSignal_b \sqsubseteq \exists parameters.(badSignal_b \sqcap available_c) \quad (9)$$
$$badSignal_b \sqsubseteq \exists parameters.(badSignal_b \sqcap unavailable) \quad (10)$$
$$badSignal_b \sqcap available_c \sqsubseteq \exists changeBearer_c.(connected_c) \quad (11)$$
$$badSignal_b \sqcap unavailable \sqsubseteq \exists disconnect.(disconnected) \quad (12)$$
$$connected_b \sqsubseteq \exists switchOff.disconnected \quad (13)$$
$$marginal_b \sqsubseteq \exists switchOff.disconnected \quad (14)$$
$$badSignal \sqsubseteq \exists switchOff.disconnected \quad (15)$$

Let us denote by UES the set of all UEs. By STATES we denote the states s_i in the BWM model. The assertion box contains one statement presenting the initial state for each UE:

$$s_0 : \sqcap_{ue \in UES} (disconnected) \quad (16)$$

To express the fact that each UE is in exactly one state at any moment we use the statement:

$$\top \sqsubseteq \neg(\sqcup_{s1,s2 \in STATES,\ s1 \neq s2}(s1 \sqcap s2)) \sqcap (\sqcup_{s \in STATES}\ s) \quad (17)$$

The UE state changes by action functions. An action function $Func_{STATES}$ for given state corresponds to the possible transitions in the BWM model. For example, the expression $Func_{STATES}(connected_b)=\{signalDrop\} \cup \{switchoff\}$ means that if the UE is connected, then the signal strength of the used bearer may drop or the UE may be switched off.

The fact that each UE can change the BWM state only by means of certain actions is represented by the following statement: for all $s \in STATES$, and all $R \notin Func_{STATES}(s)$, $s \sqsubseteq \forall R.s$.

The BWM applications may be modelled as transformations on the knowledge base using contexts $C[\varphi]$ as subformula φ of any formula ψ.

In order to describe the BWM application which selects the bearer based on the subscriber's balance additional concepts, acts and roles are defined:

- $balanceStatus_b$, the application waits for subscriber balance;
- $connectivityStatus_b$, the application waits for UE connectivity parameters;
- *lowBalance*, the subscriber balance is low;
- $cheapest_b$, the bearer *b* is the cheapest one;
- $available_c$, bearer *c* is available;
- *outOfCredit*, the subscriber is out of credit;
- *getBalance*, the application requests the status of subscriber's balance;
- *getParameters*, the application requests the UE connectivity parameters;

- *balanceIsOver*, the balance is over;
- *balanceIncrease*, the subscriber balance increases above an application defined threshold;
- *balanceDecrease*, the subscriber balance decreases under an application defined threshold;
- *balance*, the application receives the subscriber balance status;
- *parameter*, the application receives the UE connectivity parameters.

Then the application logic is defined by using contexts which change the knowledge base.

$$C_1[BWM \sqcap connected_b] \sqsubseteq \exists getBalance.C_2[balanceStatus_b] \quad (18)$$

$$C_3[BWM \sqcap balanceStatus_b] \sqsubseteq \exists balance.C_4[connected_b \sqcap \neg lowBalance] \quad (19)$$

$$C_5[BWM \sqcap balanceStatus_b] \sqsubseteq \exists balance.C_6[connected_b \sqcap lowBalance \sqcap cheapest_b] \quad (20)$$

$$C_7[BWM \sqcap balanceStatus_b] \sqsubseteq \exists balance.C_8[connected_b \sqcap lowBalance \sqcap \neg cheapest_b] \quad (21)$$

$$C_9[BWM \sqcap connected_b \sqcap lowBalance \sqcap \neg cheapest_b] \sqsubseteq \exists getParameters.C_{10}[connectivityStatus_b \sqcap lowBalance \sqcap \neg cheapest_b] \quad (22)$$

$$C_{11}[BWM \sqcap connectivityStatus_b \sqcap lowBalance \sqcap \neg cheapest_b \sqsubseteq \exists parameters.C_{12}[connected_b \sqcap available_c \sqcap cheapest_c \sqcap lowBalance] \quad (23)$$

$$C_{13}[BWM \sqcap connectivityStatus_b \sqcap lowBalance \sqcap \neg cheapest_b \sqsubseteq \exists parameters.C_{14}[connected_b \sqcap \neg available_c \sqcap cheapest_c \sqcap lowBalance) \quad (24)$$

$$C_{15}[BWM \sqcap connected_b \sqcap available_c \sqcap cheapest_c \sqcap lowBalance] \sqsubseteq \exists changeBearer_c.C_{16}[connected_c \sqcap lowBalance \sqcap cheapest_c] \quad (25)$$

$$C_{17}[BWM \sqcap connected_b] \sqsubseteq \exists balanceIsOver.C_{18}[connected_b \sqcap outOfCredit] \quad (26)$$

$$C_{19}[BW \sqcap connected_b \sqcap outOfCredit] \sqsubseteq \exists disconnect.C_{20}[disconnected] \quad (27)$$

$$C_{21}[BWM \sqcap balanceStatus_b] \sqsubseteq \exists switchOff.C_{22}[disconnected] \quad (28)$$

The logic of BWM application, which selects the bearer based on the subscriber's balance, may be described by:

$$BWM \sqsubseteq \neg(connected_b \sqcap available_c \sqcap lowBalance \sqcap cheapest_c) \quad (29)$$

Following the same approach other BWM applications may be defined. For example, let us consider an application which selects the bearer to be used by the UE based on QoS experienced by the UE (QOS application).

The following new concepts, facts and roles are defined:

- $qosStatus_b$, the application waits for QoS experienced by the UE;
- $goodQoS_b$, the QoS of bearer *b* is acceptable.
- *qosDecrease*, the QoS received by the UE decreases under an application defined threshold.

- *qos*, the application receives the QoS status.
- *getQoS*, the application queries the device about received QoS

The application logic extends the knowledge base as follows:

$$C_1[QOS \sqcap connected_b] \sqsubseteq \exists getQoS.C_2[qosStatus_b] \quad (30)$$

$$C_3[QOS \sqcap qosStatus_b] \sqsubseteq \exists qos.C_4[connected_b \sqcap goodQoS] \quad (31)$$

$$C_5[QOS \sqcap qosStatus_b] \sqsubseteq \exists qos.C_6[connected_b \sqcap \neg goodQoS] \quad (32)$$

$$C_7[QOS \sqcap connected_b \sqcap \neg goodQoS_b] \sqsubseteq \exists getParameters.C_8[connectivityStatus_b \sqcap \neg goodQoS_b] \quad (33)$$

$$C_9[QOS \sqcap connectivityStatus_b \sqcap \neg goodQoS_b] \sqsubseteq \exists parameters.C_{10}[connected_b \sqcap available_c \sqcap \neg goodQoS_b] \quad (34)$$

$$C_{11}[QOS \sqcap connected_b \sqcap available_c \sqcap \neg goodQoS_b] \sqsubseteq \exists changeBearer_c.C_{12}[connected_c] \quad (35)$$

$$C_{13}[QOS \sqcap connected_b \sqcap \neg goodQoS_b] \sqsubseteq \exists parameters.C_{14}[connected_b \sqcap unavailable_c \sqcap \neg goodQoS_b] \quad (36)$$

$$C_{15}[QOS \sqcap connected_b \sqcap unavailable \sqcap \neg goodQoS_b] \sqsubseteq \exists disconnect.C_{16}[disconnected] \quad (37)$$

$$C_{17}[QOS \sqcap connected_b \sqcap goodQoS_b] \sqsubseteq \exists qosDecrease.C_{18}[\sqcap connected_b \sqcap \neg goodQoS_b] \quad (38)$$

$$C_{19}[QOS \sqcap qosStatus_b] \sqsubseteq \exists switchOff.C_{20}[disconnected] \quad (39)$$

$$C_{21}[QOS \sqcap badSignal_b \sqcap available_c] \sqsubseteq \exists changeBearer_c.C_{22}[qosStatus_b] \quad (40)$$

The logic of QOS application, which selects the bearer based on the experienced QoS, may be described by:

$$QOS \sqsubseteq \neg(connected_b \sqcap available_c \sqcap \neg goodQoS_b) \quad (41)$$

5. REASONING ON APPLICATION INTERACTION

Having described applications as statements in the knowledge base, it is possible to reason about undesired application interaction. Application interaction is considered as satisfiability problem, i.e. applying statements in the knowledge base related to the logic of two or more applications leads to contradiction. For example, an undesired interaction between BWM application and QOS application may occurs when the subscriber balance is low, and the QoS of the cheapest bearer is not acceptable.

Applying standard reasoning to the knowledge base we derive the following sequences.

Using tableau method defined in [10] and [11], consecutive application of rule results in the following:

1. As to (16), initially UE is disconnected.
2. As to (1), the UE is switched on and connects to bearer *b*.
3. As to (18), the BWM application queries about subscriber's balance.

4. As to (30), the subscriber's balance is low and *b* is the cheapest bearer.
5. As to (20), the QOS application queries the UE about the received QoS.
6. As to (31), the QoS experienced by the UE is acceptable.
7. As to (38), the QoS experienced by the UE decreases.
8. As to (33), the QOS application queries the UE about its connectivity parameters.
9. As to (34), there is an available bearer *c*.
10. As to (35), the QOS application requires UE to change the bearer to *c*.
11. As to (30), the QOS application queries UE about experienced QoS.
12. As to (31), the QoS experienced by the UE is acceptable.
13. As to (22), the BWM application queries the UE about its connectivity parameters.
14. As to (23), the cheapest bearer is available, but not used, which contradicts to the logic of BWM application as to (25). ■

The result shows that when applying both applications to the same UE, statements representing the BWM and QOS do not satisfy the statements in the knowledge base i.e. the applications contradict to each other.

Other undesired interactions may occur between an application which selects a bearer based on UE location and another application that selects a bearer based on requested and experienced data speeds.

Once detected, interactions may be resolved by setting priorities. The MEC functionality for service orchestration determines the required behaviour in case of interaction based on application priority. Application with higher priority can override the instructions of application with lower priority.

6. CONCLUSION

The MEC *BandwidthManagement* feature allows authorized application to dynamically control the QoS available on UE connections. The application-driven QoS is achieved by policy-based connectivity management which results in intersystem handover procedures in the network. Deployment of bandwidth management functionality in the network requires interaction with Policy and Charging Control and Online Charging Control functions. In this paper, bandwidth management models representing the application and network views are studied and a method for their formal verification is proposed. Further, an abstraction of bandwidth management data is defined representing the main concepts and roles, and their relationship. The semantic annotation allows intelligent data processing, e.g. data analysis, aggregation or interpretation. The proposed semantic annotation enables re-use of bandwidth management data by many M2M applications and simplifies application configuration.

MEC service platform provides service orchestration functionality. The service orchestration needs to discover and resolve dynamically undesired application behaviour. The paper proposes an approach to service orchestration. Undesired interactions between MEC applications are considered as satisfiability problem. They may be detected by standard reasoning on the knowledge base representing bandwidth management data.

The semantic data annotation and the application logic may be described by Ontology Web Language (OWL). There exist a number of ontology editors and frameworks for constructing domain models and knowledge-based applications with ontologies and reasoners to infer logical consequences from a knowledge base.

Resolving undesired application interaction by the use of priorities allows dynamic service orchestration. The level of human involvement in the network management can be reduced due to automatic orchestration of applications provided by different service providers with coherence and consistency in order to provide adaptable service continuity to end users.

REFERENCES

[1] Corcoran, P., 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, 2016, pp. 73-74.

[2] Li, H., G. Shou, Y. Hu, Z.Guo, "Mobile Edge Computing: Progress and Challenges," *4th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2016, pp. 83-84.

[3] Ahmed, A., E. Ahmed, "A Survey on Mobile Edge Computing," *10th IEEE International Conference on Intelligent Systems and Control* (ISCO 2016), pp. 1-8, 2016.

[4] Atanasov, I., "Modeling Aspects of Autonomous Smart Metering Information Systems," *Int. Journal on Information Technologies & Security*, IJITS, vol.8, No 1, 2016, pp.3-17, ISSN 1313-8251

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

[6] Maternaghan, C., K. Turner, "Policy Conflicts in Home Automation," *Computer Networks*, vol.57 issue 12, 2013, pp.2429-2241.

[7] Lin, Y.B., et al., "EasyConnect: A Management System for IoT Devices and Its Applications for Interactive Design and Art," *IEEE Internet of Things*, vol.2, issue 6, 2015, pp. 551-561.

[8] Apel, S., C. Kastner, B. Garvin, "Exploring Feature Interactions in the wild: the new feature interaction challenge," *International Conference on Feature-Oriented Software Development, FOSD,* 2013, ACM, pp.1-8,

[9] Atanasov, I., E. Pencheva, "A Formal Approach to Service Interaction Detection in Mobile Networks," *10th WSEAS Int. Conf. on Software Engineering, Parallel and Distributed Systems* (SEPADS '11) Cambridge, UK, 2011, pp.118-123.

[10] Atanasov, I., E. Pencheva, "Reasoning on Service Interaction in Mobile networks," *International Journal of Computers and Communications*, vol.15 (2), 2011, pp.59-66.

[11] 3GPP Technical Specification Group Services and System Aspects; General Packet Radio Service (GPRS) enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) access, Release 14, v14.2.0, 2016

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

[13] 3GPP TS 23.401Technical Specification Group Services and System Aspects; Policy and charging control architecture, Release 13, v13.7.0, 2016.

[14] Sarria, D. Park, D., Jo, M. Recovery for Overloaded mobile edge computing, Future Generation Computer Systems, Available at:http://dx.doi.org/10.10.2016/ j.futu re.2016. 06.024, 2016.

[15] Deng, M., H. Tian, X. Lyu. Adaptive sequential offloading game for multi-cell Mobile Edge Computing, *23rd International Conference on Telecommunications* (ICT'2016), Greece, 2016, pp. 1-5.

[16] Katsalis, K., T. G. Papaioannou, N. Nikaein. L. Tassiulas. SLA-Driven VM Scheduling in Mobile Edge Computing, *IEEE 9th International Conference on Cloud Computing (CLOUD 2016)*, USA, 2016, pp. 750-757.

[17] Open Mobile Alliance, Diagnostics and Monitoring Trap Events Specifications," OMA-TS-DiagMonTrapEvents-V1_2-20131008-A, 2013.

[18] Escrig, D., J. Keiren, T. Willemse, "Games for Bisimulations and Abstraction", *Cornel University Library*, arXiv:1611.00401 [cs.LO]

Information about the author:

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

Manuscript received on 11 April 2017