ON FUNCTIONALITY OF POLICY CONTROL AT THE NETWORK EDGE [1];

Ivaylo Atanasov, Evelina Pencheva, Aleksandar Nametkov, Ventsislav Trifonov

Technical University of Sofia
{iia,enp}@tu-sofia.bg, aleksandar.nametkov@balkantel.net, vgt@tu-sofia.bg
Bulgaria

Abstract: Fifth generation (5G) mobile communication system has to provide advanced policy-based control functions in order to address the diverse requirements of different use cases. This paper studies the capabilities to distribute the policy control functionality of the core network in 5G at the network edge. Edge computing possesses the potential for exposing network capabilities to third party applications close to the end users. The paper presents an approach to define RESTful Application Programming Interfaces (APIs) that enable mobile edge applications to monitor the subscriber spending limits regardless the way of measurement i.e. monetary, volume, duration, etc., as well as to control sponsored connectivity. Using the proposed APIs, a mobile edge application may track the usage of network resources by a subscriber and to enforce actions related to user traffic such as quality of service downgrade, traffic blocking or redirection in the vicinity of end users. It also may set information identifying a specified application and traffic rules enabling sponsor connectivity. The proposed APIs are described by typical use cases which illustrate the API functionality, by data models which enable different applications to store and access data in an interoperable fashion, by interface definitions, and by formally verified state models which consider some API implementation issues.

Key words: Fifth Generation Mobile Communications, Policy control, Multi-access Edge Computing

1. INTRODUCTION

The fifth generation (5G) mobile communication system is positioned to address the diverse requirements of very wide service types and cloud native applications. It is expected to enable ubiquitous connectivity supporting tremendous growth in

[1] The research is part of project DH07/10-2016, funded by Bulgarian National Science Fund, Ministry of Education and Science.

volume/density of traffic. The 5G system has to improve network efficiency in order to control highly heterogeneous environment and to provide ultra reliable and low latency communications, enhanced broadband connectivity and higher mobility [1], [2], [3], [4].

One of the key technologies for enhancing network functional and architectural viability, including increased autonomy and reduced capital expenditure, is Network Function Virtualization (NFV) [5]. The NFV can address the 5G design challenges through virtualized network resources, computing and storage functionality, and service abstraction [6], [7].

The conceptual architectural framework of 5G includes both traditional cloud deployments and edge deployments [8]. Multi-access Edge Computing (MEC) is an attempt to improve modularity and scalability of the network by disaggregation of cloud capabilities in the vicinity to end users. While NFV provides flexibility by dynamic network function deployment, MEC brings intelligence at the network edge [9], [10], [11].

In this paper, we study the MEC capabilities for policy-based charging control. With existing 3GPP specifications policy and charging control functionality is a part of the 5G core network. With distributing core network functionality at the network edge it is possible to provide charging control close to the end users. This paper is an extended version of [12]. In [12], MEC APIs are proposed to track the spending limits applicable to a subscriber. In this paper, we further exploit the MEC capabilities for to enable support for sponsored connectivity at the network edge.

The rest of the paper is organized as follows. Next section presents the research motivation, discussing the benefits of distributing policy and charging control functionality at the network edge. Section 3 provides a functional description of the proposed mobile edge APIs. Section4 presents required information and data models and Section 5 provides the API definitions. In Section 6, some implementation details are discussed, including models representing the mobile edge application logic and the resource states as seen by the networks. Models are formally described and it is proved that they expose equivalent behaviour. The conclusion summarizes the contribution.

2. RESEARCH MOTIVATION

Policy and Charging Control is an important component of 5G core networks that brings together and enhances capabilities from previous generations to deliver dynamic control of policy and charging on a per subscriber and per IP flow basis [13]. It encompasses Flow Based Charging for network usage, including charging control and online credit control, for service data flows and application traffic and Policy control for session management and service data flows (e.g. gating control, quality of service (QoS) control, etc.).

Subscriber spending limits is a function that enables policy decisions based on the status of policy counters that are maintained in the Online Charging System (OCS) [14]. Policy counter is a mechanism to track spending applicable to a subscriber. The policy counter status is a label whose values are not standardized and it is associated with a policy counter's value relative to the spending limit(s). The number of possible policy counter status values for a policy counter is one greater than the number of thresholds associated with that policy counter, i.e. policy counter status values describe the status around the thresholds. This is used to convey information relating to subscriber spending from OCS [15].

We propose to relocate the subscriber spending limit function from the core network to the network edge. This will enable more timely reaction in case of reaching a threshold defined for a policy counter, i.e. policy-based decisions may be enforced close to the end user.

Some of the applications related to control on spending limits are as follows.

The operator and the customer may negotiate spending limits (e.g. volume, duration monetary) and after which the customer traffic will be automatically cut-off. Upon approaching the negotiated thresholds a warning message may be sent to the customer. Similar use case enables an agreement for higher data speeds for predefined period of time or data volume after which the user traffic is shaped.

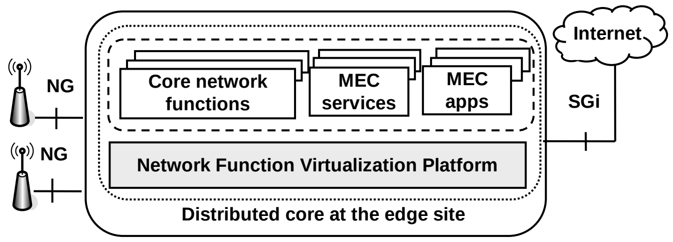
Another use case is when the customer has a prepaid data service, which allows consumption of a specified data volume. When this volume is reached, the HTTP traffic is redirected to specific application server to enable recharging in order to purchase a supplementary volume of data that may be consumed.

The control on subscriber spending limits at the network edge may be combined with PCC functionality for sponsor connectivity support [13]. For example, a streaming service provider (sponsor) offers free of charge movie trailers in order to provide the opportunity for the customers to see a tart of the movie before purchase. In this case the sponsor pays for the customer connectivity. With this feature, it has to be possible to detect the traffic of specific application and to provide limits on application usage.

Fig.1 shows the proposed architecture for deployment of policy decisions based on subscriber spending limits and support for sponsored connectivity.

Mobile edge services and applications can be co-located with virtualized core network functions on the same virtualized platform which improves scalability and results in better utilization of network resources.

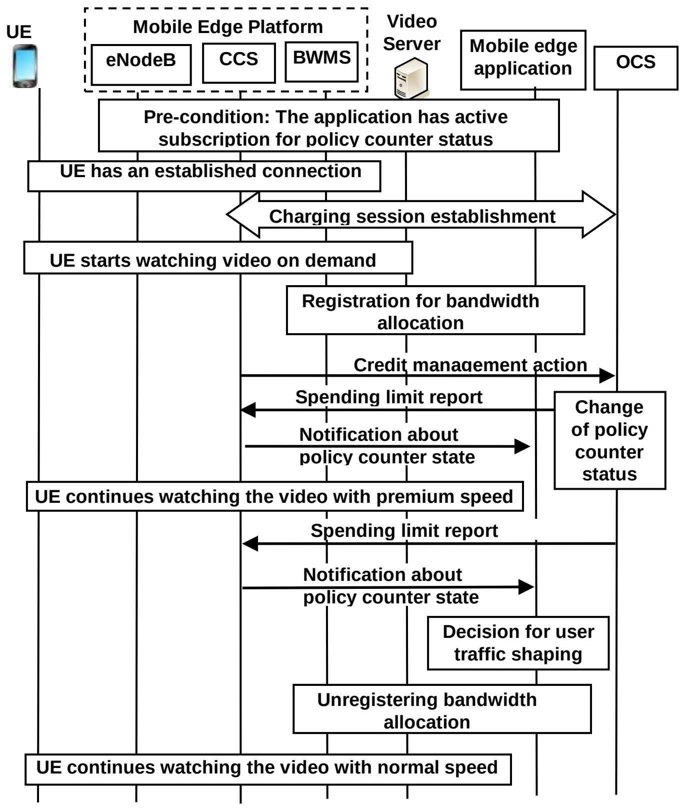
The standardized Bandwidth Management Service (BWMS) allows different mobile edge applications to request specific bandwidth requirements (bandwidth size, bandwidth priority, or both). The BWMS may aggregate all the requests and act in a manner that will help to optimize the bandwidth usage [16]. The MEC server may manipulate user traffic and may perform charging. In addition to minimization of latency, it is possible to steer the user traffic on a per session/packets bases.



Internet
NG
NG
Core network functions
MEC services
MEC apps
SGi
Network Function Virtualization Platform
Distributed core at the edge site

Fig.1. MEC and distributed core network co-location

The proposed mobile edge service for control on subscriber spending limits is called Charging Control Service (CCS).



UE
Mobile Edge Platform
eNodeB
CCS
BWMS
Video Server
Mobile edge application
OCS
Pre-condition: The application has active subscription for policy counter status
UE has an established connection
Charging session establishment
UE starts watching video on demand
Registration for bandwidth allocation
Credit management action
Spending limit report
Change of policy counter status
Notification about policy counter state
UE continues watching the video with premium speed
Spending limit report
Notification about policy counter state
Decision for user traffic shaping
Unregistering bandwidth allocation
UE continues watching the video with normal speed

Fig.2. Use case of policy based-decision on spending subscriber limits

An example use case for subscriber spending limits is illustrated in Fig.2. The example shows a scenario where a user starts watching a video on demand application. The mobile edge application registers with the BWMS the video bandwidth requirements for premium rate and the user enjoys the high quality video experience. When the user reaches a certain spending limit, the mobile edge application decides to downgrade the rate for the video stream and it deregisters the video application for premium bandwidth allocation.

Generally, the ability to monitor the subscriber spending limits is the foundation of many mobile charging plans. Distributing the policy-based functionality based on tracking the subscription spending at the network edge provides more flexibility in packet filter handling and application level charging.

We propose also an extension of BWMS enables an authorized mobile edge application to request the detection of specified application traffic and to be notified on the start and stop of application traffic. It is also possible to apply specific enforcement actions for the specified application traffic. The enforcement actions may include blocking of application traffic (gating control), bandwidth limitations (traffic shaping), and redirection of traffic to another address (e.g. to another server application server with service provisioning page).

An example of sponsored connectivity is a trailer of streaming service provided by MEC server. The proposed extension of the BWMS (EBWMS) may be used to detect the usage of the trailer from a particular User Equipment (UE) and to report this usage to authorized mobile edge application. The mobile edge application may, in turn, reserve appropriate resources using BWMS API, e.g. it will reserve certain high bandwidth for the duration of the trailer. An example of use case illustrating the proposed BWMS extension is shown in Fig.3.

3. FUNCTIONAL DESCRIPTION OF PTOPOSED MOBILE EDGE APIS

3.1. Policy Decisions on Subscriber Spending Limits

The proposed mobile edge Charging Control Service provides access to the status of policy counter(s) related for a user through:

- Request for the policy counter(s) status of a subscriber;
- Notification upon change in policy counter(s) status.

The communication between the mobile edge services and applications follows the Representational State Transfer (REST) style.

Fig.4 shows the message flow for application requesting policy counter information. For a mobile edge application to determine the status of policy counter, it sends a GET request to the CCS providing subscriber’s and policy counter’s identification. The MEC platform opens a dialogue with the OCS to retrieve information about the policy counter requested. The CCS receives a response with the status of the policy counter requested.

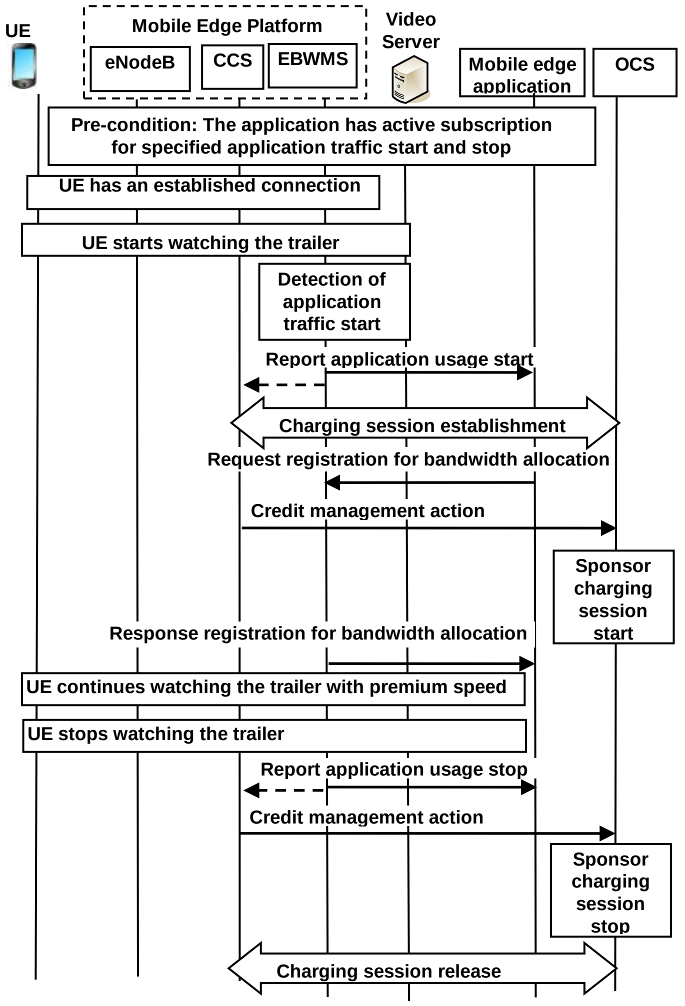

UE
Mobile Edge Platform
eNodeB
CCS
EBWMS
Video Server
Mobile edge application
OCS
Pre-condition: The application has active subscription for specified application traffic start and stop
UE has an established connection
UE starts watching the trailer
Detection of application traffic start
Report application usage start
Charging session establishment
Request registration for bandwidth allocation
Credit management action
Sponsor charging session start
Response registration for bandwidth allocation
UE continues watching the trailer with premium speed
UE stops watching the trailer
Report application usage stop
Credit management action
Sponsor charging session stop
Charging session release

Fig. 3. Use case of support for sponsored connectivity

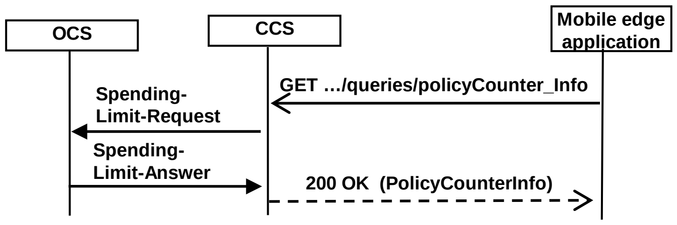


Fig.4. Flow of application requesting policy counter information

To receive notification about policy counter change, the mobile edge application creates a subscription to the related event that is available at CCS. Fig.5 shows a scenario where a mobile edge application sends POST request to the CCS to create a subscription for notifications about policy counter change. The request body contains **PolicyCounterData** data structure to the resource, representing the subscription. The application includes in **PolicyCounterData** structure the address where it wishes to receive notifications. The CCS returns a response with message body containing **PolicyCounterData** data structure. The data structure contains the address of the resource created and the subscribed event type.

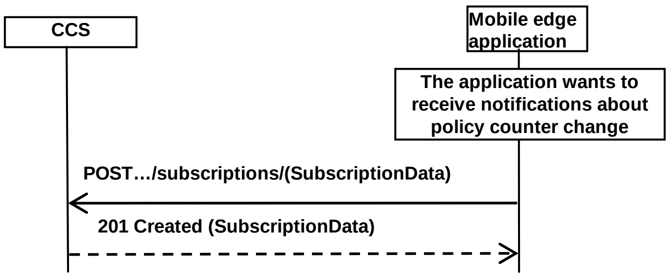

CCS
Mobile edge application
The application wants to receive notifications about policy counter change
POST…/subscriptions/(SubscriptionData)
201 Created (SubscriptionData)

Fig. 5. Flow of subscribing to the policy counter change information

CCS may define an expiry time for the subscription to policy counter events. In case expiry time is used, the time is included in the **PolicyCounterData** structure in the response message to the subscription request. On subscription expiry, CCS sends a POST request with a notification to the call-back address provided by the application that owns the subscription, as shown in Fig.6.

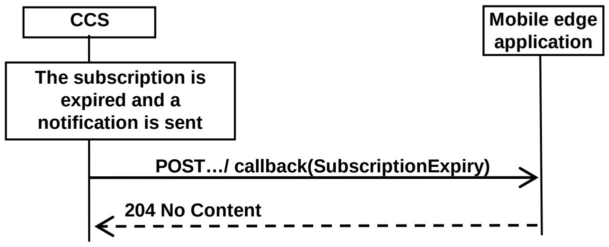

CCS
Mobile edge application
The subscription is expired and a notification is sent
POST…/ callback(SubscriptionExpiry)
204 No Content

Fig. 6. Flow of CCS sending notification on subscription expiry

In case of subscription expiry, the mobile edge application needs to update the subscription for policy counter status events. It sends a PUT request to the resource representing the subscription, as shown in Fig.7. The message body of the response contains the accepted data structure specific to that subscription.

Following the same message pattern the mobile edge application may terminate the subscription to policy counter status events. It sends a DELETE request to the resource representing the respective subscription.

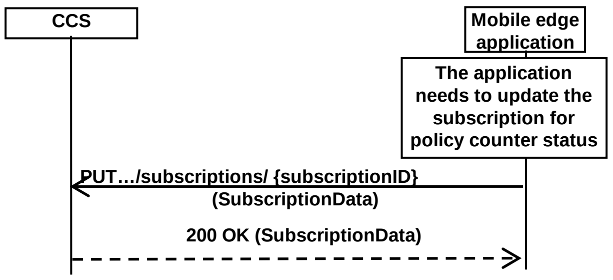

CCS
Mobile edge application
The application needs to update the subscription for policy counter status
PUT…/subscriptions/ {subscriptionID} (SubscriptionData)
200 OK (SubscriptionData)

Fig. 7. Flow of subscription modification

A mobile edge application can be notified of the status of policy counter in case of status change. Fig.8 shows the message flow for notification about policy counter status change.

When the OCS detects a status change of the policy counter of interest it reports the change to the mobile edge platform. The CCS sends a POST request with message body containing the **PolicyCounterData** data structure to the callback address provide by the application.

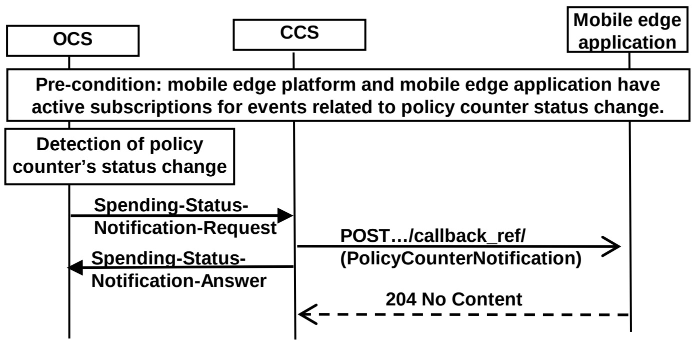

OCS
CCS
Mobile edge application
Pre-condition: mobile edge platform and mobile edge application have active subscriptions for events related to policy counter status change.
Detection of policy counter's status change
Spending-Status-Notification-Request
POST…/callback_ref/ (PolicyCounterNotification)
Spending-Status-Notification-Answer
204 No Content

Fig.8 Flow of notification about policy counter's status change

3.2. Support for Sponsor Connectivity

The extended BWMS (EBWMS) enables detection of usage start and stop of a specified application for both the mobile edge application and mobile edge platform. To be able to monitor application detection, the mobile edge application installs the traffic rules at the mobile edge platform on the specified application traffic.

Fig.9 shows a scenario where the mobile edge application installs application detection rules which request reporting on detection of application traffic start and stop.

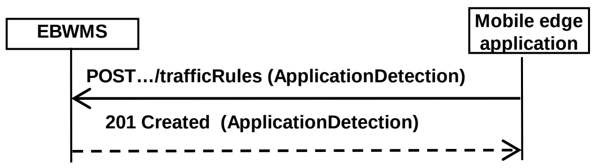

EBWMS
Mobile edge application
POST…/trafficRules (ApplicationDetection)
201 Created (ApplicationDetection)

Fig. 9. Flow of installing an application detection rule

When the mobile edge application wants to monitor application detection to specific application traffic, it installs traffic rules, requesting reporting on detection of specified application traffic start and stop.

EBWMS may define an expiry time for the application detection. In case expiry time is used, the time will be included in the {**ApplicationTrafficReport**} data structure that is included in the response message to the application detection request. Prior to the expiry, EBWMS will also send a notification to the application that owns the application detection. The mobile edge application may also update or terminated an existing application detection monitoring.

In case of installed application detection rules, the EBWMS reports to the mobile edge application on detection of application traffic start and stop. Fig.10 presents the scenario where the EBWMS sends notification about the specific application traffic start or stop to the mobile edge application.

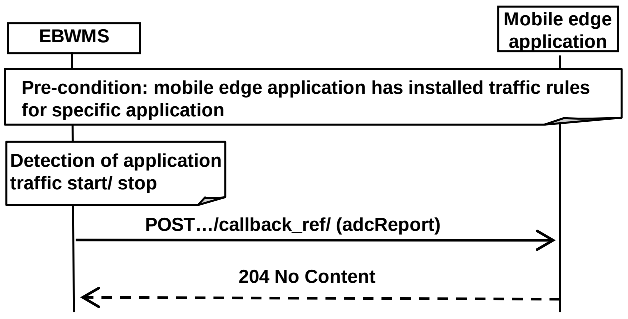

EBWMS
Mobile edge application
Pre-condition: mobile edge application has installed traffic rules for specific application
Detection of application traffic start/ stop
POST…/callback_ref/ (adcReport)
204 No Content

Fig. 10. Flow of receiving a report of detected application traffic start/stop

Upon receiving a report on application traffic start, the mobile edge application may do one of the following:

- apply gating of application traffic, i.e. blocking the application traffic to pass through to the desired endpoint;

- apply limiting the application traffic to a certain uplink or downlink maximum bit rate;

- redirect the detected application traffic to another destination.

The mobile edge APIs for user traffic handling are proposed in [17], [18].

4. DATA MODELS

This section describes the data models of resources representing the access to subscriber spending limits functionality and for support of sponsored connectivity functionality.

The **PolicyCounterInfo** data type represents information about the status of policy counters that are associated with a specific mobile edge application instance. The attributes of **PCStatus** are as follows:

- **timeStamp** identifies when the policy counter status change occurred;
- **appInsId** uniquely identifies the mobile edge application instance;
- **requestId** uniquely identifies the request for the policy counter status information. It is allocated by the application;
- **userID** uniquely identifies the subscription of interest;
- **policyCounterList** is a structure of one or more **policyCounter** and indicates the list of policy counter identifiers to be subscribed to;
- **policyCounter** represents information about the policy counter status. It is a structure of **policyCounterID, policyCounterStatus,** and **pendingPolicy-CounterInfo**;
- **policyCounterID** uniquely identifies a policy counter of interest;
- **policyCounterStatus** identifies the policy counter status applicable for the subscriber. The actual values are not specified, but an example values are provided in the next section;
- **pendingPolicyCounterInfo** contains the pending counter status and the active time. It is a structure of **policyCounterStatus,** and **pendingPolicy-CounterChangeTime;**
- **pendingPolicyCounterChangeTime** indicates the expected time at which the pending policy counter status becomes the current status of the policy counter.

The **SubscriptionData** type represents a subscription to policy counter status change notifications from CCS. The attributes of **PCSsubscription** type are as follows:

- **callbackReference** is a URI (Uniform Resource Identifier) provided by the application, showing the place she wants to receive notifications;
- **filterCriteria** represents a list of filtering criteria for the subscription for policy counter status related events, which are also included in the response. It is a structure of **appInsId, userID,** and **policyCounterList** as defined earlier;
- **expiryDeadline** indicates when the subscription for policy counter status changes expires.

The **PolicyCounterNotification** type represents a notification about policy counter status change. It is a list of **policyCounterStatus.**

The **ApplicationDetection** type represents a monitoring on application traffic start and/or stop. The attributes of **ApplicationDetection** are as follows:

- **callbackReference** as described above;
- **filterCriteria** represents a list of filtering criteria for the application detection. Any filtering criteria from below, which are included in the request, shall also be included in the response. It is a structure of **appInsId, appDetId,** and **adEvent**;
- **appInsId** represents a unique identifier for the mobile edge application instance and it is of String type;
- **appDetId** references the application for which the application detection applies;
- **adEvent** determines the event upon which a report has to be sent. It is of Enumerated type:

0=Application_Traffic_Start; 1=Application_Traffic_Stop; 2=Application_Traffic_Start_and_Stop;

- **expiryDeadline** is of TimeStamp type.

The **expiryMonitoring** type represents a notification from the EBWMS with regards to expiry of the existing application usage monitoring. The notification is sent by the EBWMS to inform the mobile edge application about expiry of the application usage monitoring. The attributes of this type are **timestamp** and **expiryDeadline**.

The data types related to application detection report include the following:

- **adReport** is used by the mobile edge platform to provide information about detected specified application traffic to the mobile edge application. It is a structure of **timestamp, 5gduti, appDetId** and **event**;
- **5Gguti** is an unambiguous identification of the UE that does not reveal the UE or the user's permanent identity in the 5G System. It also allows the identification of the Access and Mobility Management Function (AMF) and network. This is a structure of **mcc, mnc, amfID** and **5Gtmsi;**
- **mcc** is the Mobile Country Code;
- **mnc** is the Mobile Network Code;
- **amfID** is the identification of AMF;
- **5Gtmsi** is the Temporary Mobile Subscriber Identity;
- **event** is of Enumerated type and it indicates the reason for reporting: 0=Application_Traffic_Start; 1= Application_Traffic_Stop.

The proposed data models enable interoperability as it provides uniform access to the same data structure by different applications.

5. INTERFACE DEFINITION

The structure of resources, supported by the CCS, is shown in Fig.11. Each resource has a unique URI. All CCS resources have the following root:

{apiRoot}/ccs/{apiVersion}/

Following the RESTful architectural style, all resources are manipulated using four operations implemented by HTTP requests: POST, GET, PUT and DELETE. Table 1 represents the resources and the supported methods.

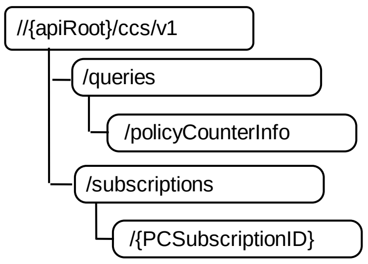


Fig. 11. Structure of resources supported by CCS

Table 1. CCS resources and supported HTTP methods

Resource name	***Resource URI***	***HTTP method***	***Description***
All queries about policy counter status	*/queries*	*GET*	*Retrieves the list of all queries about policy counter state.*
Policy counter status information	*/queries/ policyCounterInfo*	*GET*	*Retrieves information about policy counter status.*
All subscriptions for subscriber	*/subscriptions*	*GET*	*Retrieves a list with all subscriptions related to policy counter status change.*
		POST	*Creates a new subscription for policy counter status change*
Existing subscription	*/subscriptions/ subscriptionID*	*GET*	*Retrieves information about existing subscription for policy counter status change.*
		PUT	*Modifies existing subscription.*
		DELETE	*Deletes existing subscription.*

All resource URIs of the extended Bandwidth Management API have the following root:

{apiRoot}/ebwm/{apiVersion}/

Fig.12 illustrates the resource structure of the proposed EBWMS API.

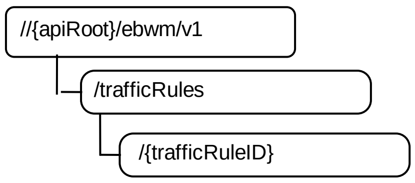


Fig. 12. Resource structure of the proposed EBWMS API

The **trafficRules** resource represents all application detection rule instances. The HTTP GET method retrieves a list of active application detection rules. The HTTP POST method creates a new application detection rule instance by sending a data structure, where the request body includes the **ApplicationDetection** data type. Upon success, a response body contains respectively **ApplicationDetection** data type.

The **trafficRuleID** resource represents existing application detection rule instance. The GET method retrieves information on current specific application detection rule. The PUT method modifies existing application detection rule instance by sending a new data structure, where the request body includes the **ApplicationDetection** data type. The DELETE method cancels the existing application detection rule instance.

Some implementation aspects of the proposed API are considered in the next Section.

5. STATE MODELS

Implementation of the mobile edge CCS and a mobile edge application that make used of CCS API requires development of models, representing the resource state. The models representing the states related to subscriber spending limits supported by the CCS and by the application need to be synchronized.

Fig.13 shows a simplified model of the application view on the policy counter status.

A simple 4 level model related to policy counter state might be defined as: **valid, pending_invalid, invalid**, and **pending_valid**. In **valid** state, the subscriber has not reached the agreed spending limit. In **pending_invalid** state, the subscriber approaches the agreed spending limit threshold and the expected time, after which this threshold is expected to be reached, is **Δt.** For example, the subscriber is allowed to use higher data speeds in less busy periods and such a period is about to expire. In **invalid** state, the subscriber has reached the agreed threshould for the spending limit. In **pending_valid** state, the subscriber has reached the threshold related to the spending limit, but the expected time for the policy counter status to become **valid** is **Δt.** In **pending_invalid** state, the subscriber may recharge his account and the policy counter status becomes **valid**. For example, the busy period during which the subscriber is not allowed to use higher data speeds is about to expire. The transitions from **valid** to **invalid** directly or vice versa are possible due to administrative actions. In **unknown** state, the application does not have information about policy counter status. In **valid** and **invalid** states, the application may subscribe for notifications about policy counter status change, as well as to terminate the subscription.

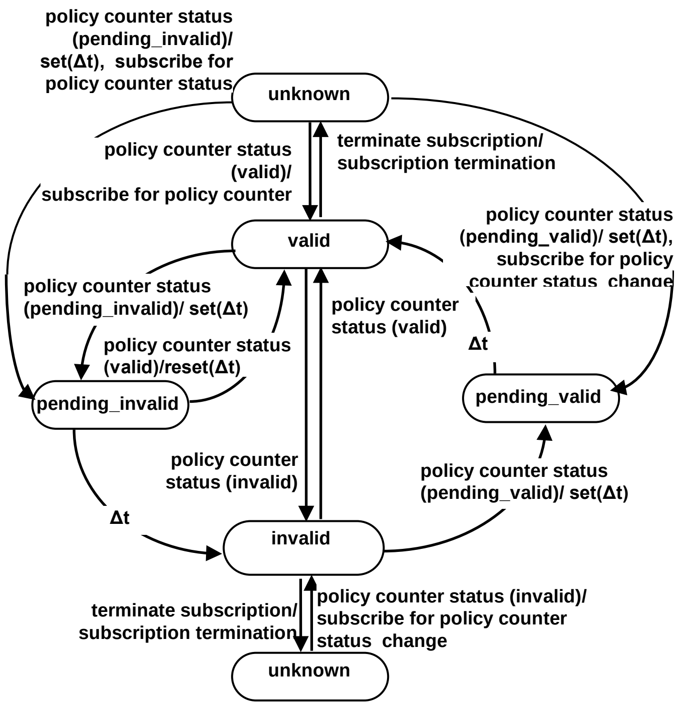

policy counter status (pending_invalid)/ set(Δt), subscribe for policy counter status
unknown
policy counter status (valid)/ subscribe for policy counter
terminate subscription/ subscription termination
policy counter status (pending_valid)/ set(Δt), subscribe for policy counter status change
valid
policy counter status (pending_invalid)/ set(Δt)
policy counter status (valid)/reset(Δt)
policy counter status (valid)
Δt
pending_invalid
pending_valid
policy counter status (invalid)
policy counter status (pending_valid)/ set(Δt)
Δt
invalid
terminate subscription/ subscription termination
policy counter status (invalid)/ subscribe for policy counter status change
unknown

Fig. 13. Model of policy counter state as seen by the mobile edge application

Fig.14 shows the model representing the tracking of spending limits for given subscriber, supported by the CCS.

In **Idle** state, the tracking of subscriber spending limits is not activated. In Idle state, the CCS may be asked about policy counter status. The mobile edge platform initiates a session with OCS (not shown in the figure), and retrieves the status of the policy counter requested. When the application subscribes for notifications related with changes in the policy counter state, it waits for changes in **WaitForPolicyCounterChange** state.

Both models are simplified; they represent only successful execution of the relevant procedures in the network and do not take into account abnormal conditions associated with unsuccessful procedures.

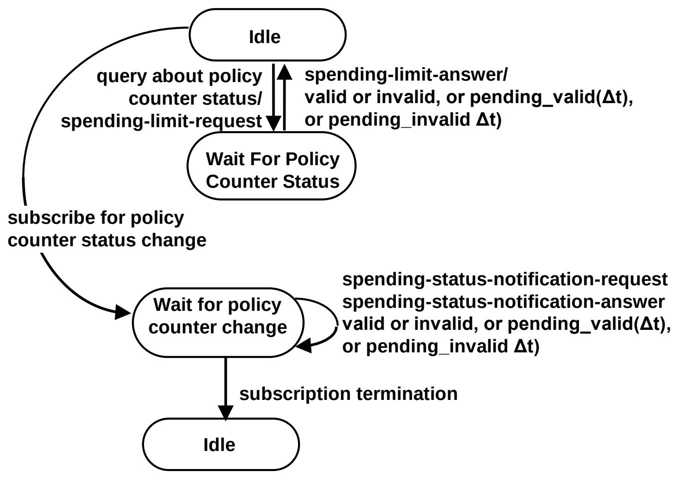

Idle
query about policy counter status/ spending-limit-request
spending-limit-answer/ valid or invalid, or pending_valid(Δt), or pending_invalid Δt)
Wait For Policy Counter Status
subscribe for policy counter status change
Wait for policy counter change
spending-status-notification-request
spending-status-notification-answer
valid or invalid, or pending_valid(Δt), or pending_invalid Δt)
subscription termination
Idle

Fig. 14. Model, representing the spending limits tracking of given subscriber, supported by the CCS

In order to prove in a mathematical manner that both models expose equivalent behaviour, we formalize the models' description. The notion of Labelled Transition System (LTS) is used to describe each model.

A Labelled Transition System is represented as quadruple of a set of states, a set of actions, a set of transitions and a set of initial states.

By $\mathrm{T}_{\mathrm{App}} = (S_{\mathrm{App}}, Act_{\mathrm{App}}, \rightarrow_{\mathrm{App}}, s_0^{\mathrm{App}})$ it is denoted an LTS, representing the model of policy counter state as seen by the application, where:

- S_{App} = {Unknown [s_1^A], Valid [s_2^A], Pending_Invalid [s_3^A], Invalid [s_4^A], Pending_Valid [s_5^A]};
- Act_{App} = {policyCounterStatus(valid)[t_1^A], policyCounterStatus(pending_invalid) [t_2^A], policyCounterStatus(invalid) [t_3^A], policyCounterStatus(pending_invalid) [t_4^A], terminateSubscription [t_5^A], Δt [t_6^A]};
- $\rightarrow_{\mathrm{App}}$ = {$(s_1^A\ t_1^A\ s_2^A)$, $(s_1^A\ t_2^A\ s_3^A)$, $(s_1^A\ t_3^A\ s_4^A)$, $(s_1^A\ t_4^A\ s_5^A)$, $(s_2^A\ t_2^A\ s_3^A)$, $(s_3^A\ t_1^A\ s_2^A)$, $(s_3^A\ t_6^A\ s_4^A)$, $(s_2^A\ t_3^A\ s_4^A)$, $(s_4^A\ t_1^A\ s_2^A)$, $(s_4^A\ t_4^A\ s_5^A)$, $(s_5^A\ t_6^A\ s_2^A)$, $(s_2^A\ t_5^A\ s_1^A)$, $(s_4^A\ t_5^A\ s_1^A)$};

s_0^{App} = { s_1^A }.

Short notations for states and actions are given in brackets.

By $T_P = (S_P, Act_P \rightarrow_P, s_0^P)$ it is denoted an LTS, representing the model for spending limits tracking of given subscriber, supported by the CCS, where:

- S_P = {Idle [s_1^P], WaitForPolicyCounterStatus [s_2^P], WaitForPolicyCounterChange [s_3^P]};
- Act_P = { queryPolicyCounterStatus [t_1^P], spendingLimitAnswer [t_2^P], subscribeForPolicyCounterChange [t_3^P], spendingStatusNorificationRequest [t_4^P], subscriptionTermination [t_5^P]};
- $\rightarrow_P$ = $\{(s_1^P\ t_1^P\ s_2^P), (s_2^P\ t_2^P\ s_1^P), (s_1^P\ t_3^P\ s_3^P), (s_3^P\ t_4^P\ s_3^P), (s_3^P\ t_5^P\ s_1^P)\}$;

s_0^P = $\{ s_1^P \}$.

The synchronized behaviour of both models is formally proved by using the concept of weak bisimilarity. Intuitively, in terms of observed behaviour, two LTSs are equivalent, i.e. they are bisimilar, if one LTS displays a final result and the other LTS displays the same result [19]. In practice, strong bisimilarity puts strong conditions for equivalence which are not always necessary. In weak bisimilarity, internal transitions can be ignored.

Proposition 1: T_{App}, and T_P are weakly bisimilar.

Proof 1: As to definition of weak bisimulation, it is necessary to identify a relation between the states of both LTSs, such as for any transition from a state in one LTS there are respective transitions from states in the other LTS.

By U_{AppP} it is denoted a relation between the states of T_{App}, and T_P, where U_{AppPe}= $\{(s_1^A, s_1^P), (s_2^A, s_3^P), (s_4^A, s_3^P), (s_2^A, s_1^P), (s_3^A, s_1^P), (s_4^A, s_1^P), (s_5^A, s_1^P)\}$. Then the following transitions for the states in U_{AppPe} are identified:

1. The mobile edge application makes a query about policy counter status and subscribes for changes in the policy counter state. The policy counter status is **valid**. For $(s_1^A\ t_1^A\ s_2^A) \exists (s_1^P\ t_1^P\ s_2^P), (s_2^P\ t_2^P\ s_1^P), (s_1^P\ t_3^P\ s_3^P)$.
2. The mobile edge application makes a query about policy counter status and subscribes for changes in the policy counter state. The policy counter status is **pending_invalid** and becomes **valid**. For $(s_1^A\ t_2^A\ s_3^A), (s_3^A\ t_1^A\ s_2^A) \exists (s_1^P\ t_1^P\ s_2^P)$, $(s_2^P\ t_2^P\ s_1^P), (s_1^P\ t_3^P\ s_3^P), (s_3^P\ t_4^P\ s_3^P)$.
3. The mobile edge application makes a query about policy counter status and subscribes for changes in the policy counter state. The policy counter status is **pending_invalid** and becomes **invalid**. For $(s_1^A\ t_2^A\ s_3^A)$, $(s_3^A\ t_6^A\ s_4^A)$ $\exists$ $(s_1^P\ t_1^P\ s_2^P), (s_2^P\ t_2^P\ s_1^P), (s_1^P\ t_3^P\ s_3^P)$.
4. The mobile edge application makes a query about policy counter status and subscribes for changes in the policy counter state. The policy counter status is **invalid**. For $(s_1^A\ t_3^A\ s_4^A) \exists (s_1^P\ t_1^P\ s_2^P), (s_2^P\ t_2^P\ s_1^P), (s_1^P\ t_3^P\ s_3^P)$.

5. The mobile edge application makes a query about policy counter status and subscribes for changes in the policy counter state. The policy counter status is **pending_valid** and becomes **valid**. For $(s_1^A\, t_4^A\, s_5^A)$, $(s_5^A\, t_6^A\, s_2^A) \exists$ $s_1^P\, t_1^P\, s_2^P)$, $(s_2^P\, t_2^P\, s_1^P)$, $(s_1^P\, t_3^P\, s_3^P)$.
6. The policy counter status is **valid** and changes to **pending_invalid** and then to **invalid**: For $(s_2^A\, t_2^A\, s_3^A)$, $(s_3^A\, t_6^A\, s_4^A) \exists (s_3^P\, t_4^P\, s_3^P)$.
7. The policy counter status is **valid** and changes to **pending_invalid** and then back to **valid**: For $(s_2^A\, t_2^A\, s_3^A)$, $(s_3^A\, t_1^A\, s_2^A) \exists (s_3^P\, t_4^P\, s_3^P)$.
8. The policy counter status is **valid** and changes to **invalid**: For $(s_2^A\, t_3^A\, s_4^A) \exists$ $(s_3^P\, t_4^P\, s_3^P)$.
9. The policy counter status is **invalid** and changes to **valid**: For $(s_4^A\, t_1^A\, s_2^A) \exists$ $(s_3^P\, t_4^P\, s_3^P)$.
10. The policy counter status is **invalid** and changes to **pending_valid**, and then to **valid**: For $(s_4^A\, t_4^A\, s_5^A)$, $(s_5^A\, t_6^A\, s_2^A) \exists (s_3^P\, t_4^P\, s_3^P)$.
11. The application terminates the subscription for policy counter status changes while the policy counter status is **valid**. For $(s_2^A\, t_5^A\, s_1^A) \exists (s_3^P\, t_5^P\, s_1^P)$.
12. The application terminates the subscription for policy counter status changes while the policy counter status is **invalid**. For $(s_4^A\, t_5^A\, s_1^A) \exists (s_3^P\, t_5^P\, s_1^P)$.

Therefore T_{App}, and T_{P} are weakly bisimilar, i.e. they expose equivalent behaviour. ■

Fig.15 shows a simplified model of application view on the EBWMS status.

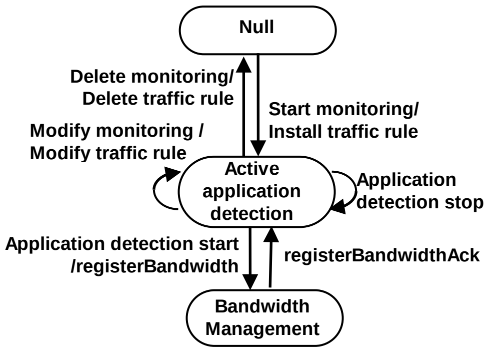


Fig. 15. Model representing the application detection status as seen by the mobile edge application for specified application traffic

In **Null** state, no traffic rule is installed. In **ActiveApplicationDetection** state, there is a traffic rule installed for specified application traffic. Upon receiving a notification about specified application traffic start, the mobile edge application registers with the EBWMS to apply specific bandwidth for the specified application. In **BandwidthManagement** state, the mobile edge application waits for acknowledge from the EBWMS.

The transitions in the model are triggered by the mobile edge application logic and application detection reports from the mobile edge platforms.

The simplified model representing the EBWMS view on the state of the application detection status is shown in Fig.16.

The model considers the RAB (Radio Access Bearers) management procedures, initiated by the network and specified application traffic monitoring.

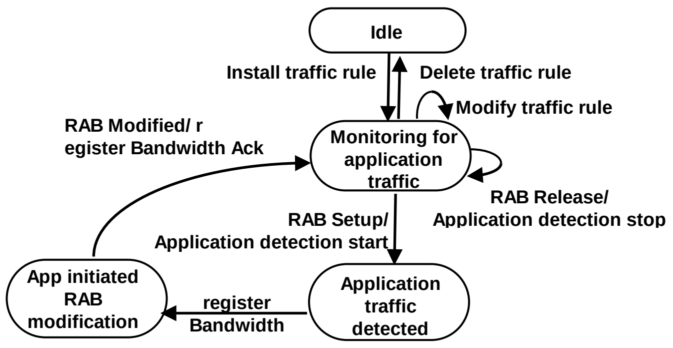

Idle
Install traffic rule
Delete traffic rule
Modify traffic rule
RAB Modified/ r egister Bandwidth Ack
Monitoring for application traffic
RAB Release/ Application detection stop
RAB Setup/ Application detection start
App initiated RAB modification
register Bandwidth
Application traffic detected

Fig. 16. Model representing the application detection status as seen by the mobile edge platform for specified application traffic

By $EBWM_{\mathrm{App}}= (S_{\mathrm{App}}', Act_{\mathrm{App}}', \rightarrow_{\mathrm{App}}', s_0^{\mathrm{App}}{}')$ it is denoted a LTS representing the mobile edge application's view on application detection status where:

- S_{App}' = {Null [s_1^{A1}], ActiveApplicationDetection [s_2^{A1}], BandwidthManagement [s_3^{A1}]};

- Act_{App}' = {startMonitoring [t_1^{A1}], modifyMonitoring[t_2^{A1}], deleteMonitoring [t_3^{A1}], applicationDetectionStart [t_4^{A1}], applicationDetectionStop [t_5^{A1}], registerBandwidthAck [t_6^{A1}]};

- $\rightarrow_{\mathrm{App}}'$ = $\{(s_1^{A1}\ t_1^{A1}\ s_2^{A1}), (s_2^{A1}\ t_2^{A1}\ s_2^{A1}), (s_2^{A1}\ t_3^{A1}\ s_1^{A1}), (s_2^{A1}\ t_4^{A1}\ s_3^{A1}),$
$(s_3^{A1}\ t_6^{A1}\ s_2^{A1}), (s_2^{A1}\ t_5^{A1}\ s_2^{A1})\}$.

$s_0^{\mathrm{App}}{}' = \{ s_1^{A1} \}$.

By $EBWM_{\mathrm{P}}= (S_{\mathrm{P}}, Act_{\mathrm{P}}, \rightarrow_{\mathrm{P}}, s_0^{\mathrm{P}})$ it is denoted a LTS representing the mobile edge application's view on ADC status where:

- S_P = {Idle [s_1^{P1}], MonitoringForAppTraffic [s_2^{P1}], ApplicationTrafficDetected [s_3^{P1}], ApplicationInitiatedRABMobification [s_4^{P1}]};
- $Act_{P'}$ = { installTrafficRule[t_1^{P1}], modifyTrafficRule [t_2^{P2}], deleteTrafficRule [t_3^{P1}], RABsetup[t_4^{P1}], bandwidthRegistration [t_5^{P1}], RABmodified [t_6^{P1}], RABrelease[t_7^{P1}]};
- $\rightarrow_P$ = {(s_1^{P1} t_1^{P1} s_2^{P1}), (s_2^{P1} t_3^{P1} s_1^{P1}), (s_2^{P1} t_2^{P1} s_2^{P1}), (s_2^{P1} t_4^{P1} s_3^{P1}), (s_3^{P1} t_5^{P1} s_4^{P1}), (s_4^{P1} t_6^{P1} s_2^{P1}), (s_2^{P1} t_7^{P1} s_2^{P1})}.

$s_0^{P'}$ = { s_1^{P1} }.

Proposition 2: The $EBWM_{App}$ and $EBWM_P$ are weakly bisimilar.
Proof 2: Let use denote by $U_{AppPlatform}$ a relation between the states of $EBWM_{App}$ and $EBWM_P$, where

$U_{AppPlatform}$ = {(s_1^{A1}, s_1^{P1}), (s_2^{A1}, s_2^{P1}), (s_3^{A1}, s_4^{P1})}.

The following transition mapping may be identified:

1. The mobile edge application installs an traffic rule and the mobile edge platform starts specified application traffic monitoring: for (s_1^{A1} t_1^{A1} s_2^{A1}) ∃ (s_1^{P1} t_1^{P1} s_2^{P1}).
2. The mobile edge application modifies the installed traffic rule: for (s_2^{A1} t_2^{A1} s_2^{A1}) ∃ (s_2^{P1} t_2^{P1} s_2^{P1}).
3. The mobile edge application deletes the traffic rule and the mobile edge platform terminates the monitoring: for (s_2^{A1} t_3^{A1} s_1^{A1}) ∃ (s_2^{P1} t_3^{P1} s_1^{P1}).
4. The mobile edge application is notified about the start of specified application traffic and it registers the specified application for specific bandwidth: for (s_2^{A1} t_4^{A1} s_3^{A1}) ∃ (s_2^{P1} t_4^{P1} s_3^{P1}), (s_3^{P1} t_5^{P1} s_4^{P1}).
5. The mobile edge application is notified about successful bandwidth registration: for (s_3^{A1} t_6^{A1} s_2^{A1}) ∃ (s_4^{P1} t_6^{P1} s_2^{P1}).
6. The mobile edge application is notified about specified application traffic stop: for (s_2^{A1} t_5^{A1} s_2^{A1}) ∃ (s_2^{P1} t_7^{P1} s_2^{P1}).

Therefore the $EBWM_{App}$ and $EBWM_P$ are weakly bisimilar. ■

The concept of bisimilarity is used to prove in a mathematically formalized manner that the approach is consistently implementable. Mathematical formalism for equivalence of behaviour is applied to generate model-based test situations in order to demonstrate compliance of a system's implementation with its specification.

6. CONCLUSION

Advanced policy control capabilities in 5G system enable monitoring the usage limit (e.g. monetary, volume, duration) that a subscriber is allowed to use. When the subscriber spending limit is reached a pre-set limit, the system is able to trigger a QoS downgrade and/or restrict access to one, several or all IP services based on operator pre-defined thresholds. When the subscriber spending limit is increased, the system is able to modify resources (e.g. QoS, bandwidth, access) to services accordingly.

The support of sponsored connectivity enables detection of specified application traffic for which the sponsor may pay.

In this paper we propose an approach to define APIs that provide open access to monitoring the subscriber spending limits and sponsored connectivity control. Using the APIs a mobile edge application may apply policy based decision for user traffic. The APIs are described by typical use cases, which illustrate policy based functionality, by data models representing the resource structure and the respective API definition.

We also propose an extension of existing BWMS with application detection function that may be used set information identifying the application and setting traffic rules for usage of specified application traffic. The proposed extension is described by data model and API definition.

Some implementation aspects are discussed concerning modelling the behaviour of the mobile edge platform and the generic logic of mobile applications using the proposed APIs.

Moving the policy-based control based on subscriber spending limits and control of sponsored connectivity at the network edge enables more flexible charging capabilities close to the end user.

REFERENCES

[1] Morocho M. E., Cayamcela, W. Lim, Artificial Intelligence in 5G Technology: A Survey, *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, Jeju, South Korea, 2018, pp. 860-865.

[2] Pandi, V. S,. J. L. Priya, A survey on 5G mobile technology, *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, Chennai, 2017, pp. 1656-1659.

[3] Zhang, P., X. Yang, J. Chen, Y. Huang, A survey of testing for 5G: Solutions, opportunities, and challenges, *China Communications*, vol. 16, no. 1, 2019, pp. 69-85.

[4] Aljiznawi, R. A., N. H. Alkhazaali, S. Q. Jabbar, .D. J. Kadhim, Quality of Service (QoS) for 5G, *International Journal of Future Computer and Communication*, vol. 6, no. 1, March 2017, pp.27-30.

[5] Rodriguez, F. L. U. S. Dias, D. R. Campelo, R. O. Albuquerque, S-J. Lim, L. G. Villalba. QoS Management and Flexible Traffic Detection Architecture for 5G Mobile Networks, *Sensors*, 2019, issue 19, vol.1335, pp. 1-21.

[6] Yousaf, F. Z., M. Bredel, S. Schaller, F. Schneider, NFV and SDN - Key Technology Enablers for 5G Networks, *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, Nov. 2017, pp. 2468 – 2478.

[7] Sinh, D., L. V. Le, L.P. Tung, B.S. P. Lin, The Challenges of Applying SDN/NFV for 5G & IoT, *IEEE Vehicular Technology Society Asia Pacific Wireless Communications Symposium, APWCS'2017*, Incheon, South Korea, pp.1-6.

[8] Lin, Y. Keynote topic: Network cloudification: SDN-NFV and 5G-MEC with edge and fog computing, *27th International Telecommunication Networks and Applications Conference (ITNAC)*, Melbourne, VIC, 2017, pp. 1-8.

[9] Sciancalepore, V., F. Giust, K. Samdanis, Z. Yousaf, A double-tier MEC-NFV architecture: Design and optimization, *2016 IEEE Conference on Standards for Communications and Networking CSCN*, Berlin, 2016, pp. 1-6.

[10] Nam, Y., S. Song, J. Chung, Clustered NFV Service Chaining Optimization in Mobile Edge Clouds, *IEEE Communications Letters*, vol. 21, no. 2, Feb. 2017, pp. 350-353.

[11] Song, S., J. Chung, Sliced NFV service chaining in mobile edge clouds, *19th Asia-Pacific Network Operations and Management Symposium, APNOMS*, Seoul, 2017, pp. 292-294.

[12] Atanasov, I., E. Pencheva, A. Nametkov, V. Trifonov, Mobile Edge Service for Charging Control, *Open Innovation Association 24th FRUCT (Finnish-Russian University Cooperation in Telecommunications)* 2019, Moscow, Russia, 4-12 April 2019, pp.17-23.

[13] 3GPP TS 23.203 Technical Specification Group Services and System Aspects; *Policy and Charging Control Architecture*, 2018, Release 15, v15.3.0.

[14] 3GPP TS 29.219 Technical Specification Group Services and System Aspects; Service aspects; *Charging and Billing*, Release 16, v16.1.0, 2018.

[15] 3GPP TS 29.219 Technical Specification Group Services and System Aspects; *Policy and Charging Control: Spending Limit Reporting over Sy reference point*, 2018, Release 15, v15.1.0.

[16] ETSI GS MEC 015. *Mobile Edge Computing (MEC); Bandwidth Management API*, v1.1.1, 2017

[17] Pencheva, E., I. Atanasov, Usage Monitoring Control in Radio Access Network, *23rd Conference of Open Innovations Association FRUCT*, Bologna, Italy, 2018, pp.306-314.

[18] Pencheva, E., I. Atanasov, D. Velkova, V. Trifonov, Application Level User Traffic Control at the Mobile Network Edge, *Open Innovation Association 24th FRUCT (Finnish-Russian University Cooperation in Telecommunications)* 2019, Moscow, Russia, 4-12 April 2019, pp.312-327.

[19] Chen, X. J., R. De Nicola, Algebraic characteristics of trace and decorated trace equivalences over tree-like structures, *Theoretical Computer Science*, 254, Elsevier, 2001, pp.337-361.

Information about the authors:

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

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

Alexander Nametkov is with the Faculty of Telecommunications, Technical University of Sofia. Currently, he works as a director of Railway Department of Balkantel ltd and his scientific research covers mission critical communications.

Ventsislav Trifonov is with the Faculty of Telecommunications, Technical University of Sofia. He received his PhD degree in security systems. Currently, he is Associate Professor and his scientific research area covers big data analytics and autonomic behaviour.

Manuscript received on 09 July 2019