Multiparty Call Control at the Network Edge

Ivaylo I. Atanasov, Evelina N. Pencheva*, Denitsa L. Velkova, Ivaylo P. Asenov
Faculty of Telecommunications, Technical University of Sofia,
Kliment Ohridski Blvd. 8, 1000 Sofia, Bulgaria
enp@tu-sofia.bg

***Abstract*—Network programmability is a key feature of fifth generation (5G) system which, in combination with cloud-based services, can support many use cases, including mission critical and healthcare communications. Programmability enables flexibility in customization of service connectivity. Multi-access Edge Computing (MEC) services and applications are enablers for network programmability. In this paper, MEC capabilities for programmability of multiparty multimedia call control at the network edge are studied. Multiparty video calls are one of the key applications of 5G, and are efficient way to exchange ideas, knowledge, expertise, information, and so on. The paper presents an approach to design MEC Application Programming Interfaces (APIs) which enable third party applications to create multiparty multimedia sessions and dynamically manage session participations. The API functionality is described by required information and message flows. The paper specifies the proposed MEC API with data model. Feasibility study includes modelling and formal validation of multiparty session state models supported by the network and mobile edge application. The latency injected by the API is evaluated by emulation.**

***Index Terms*—Next generation networking; Multi-access edge computing; Network function virtualization; Application programming interfaces.**

I. INTRODUCTION

Fifth generation (5G) system has huge potential to improve our daily lives in various aspects, including numerous mission critical and healthcare scenarios. In these scenarios, multiparty communications are an important feature, and 5G technologies can add value enabling enhanced broadband connections, ultra-low latency, and high reliability. For mission critical multiparty communications, 5G can reduce end-to-end latency, provide ultra-high reliability, and improve service coverage, and for healthcare multiparty communications, 5G can trigger optimal quality of service enforcement.

Traditionally mission critical multiparty communications include Push-To-Talk Over Cellular services used by Public Safety Agencies, such as fire brigade, police, and ambulance. However, mission critical voice services can bring advantages and for other industries, such as utilities, transport, mining, gas, and oil industries, etc. Mission critical video can enrich voice multiparty communication enabling video sharing among the multiparty members, and thus improving the perception of the conditions in the critical situation.

Health multiparty communications can be useful in diagnosis, treatment or prevention of diseases or other conditions, including stress, mental disorder, depression or health endangering environment. Examples of mobile applications intended for use in health multiparty communications can be found in [1].

Mission critical and machine type communications usually exploit dedicated networks where the communications with telecom operator core network is optional. These deployments are regarded as distributed core network functionality and enable more efficient provisioning of network intelligence, and improvement of customer experience and network performance [2], [3]. The distributed core network functions can be built on purpose using Network Function Virtualization (NFV) and are typically deployed at the network edge [4], [5].

Further increase of network intelligence at the mobile edge can be achieved by deployment of Multi-access Edge Computing (MEC). MEC provides computing environment for running cloud-based applications. MEC can address challenges imposed by mission critical and healthcare communications as it enables building of vertical segments and service deployment at network edge. Moving the cloud intelligence in the vicinity of end users reduces latency, optimizes network resource utilization, and improves security [6], [7].

In this paper, we study the capabilities for programmability of multiparty multimedia communication control using MEC technology. The focus is on Application Programming Interface (API) that enables mobile edge applications to create multiparty multimedia sessions to manage dynamically the participants involved and to control the media types for each participant. Among the others, the API supports mission critical communications, including healthcare scenarios, taking advantages of MEC.

3GPP addresses service requirements, architecture, and protocols for mission critical Push-to-Talk service in [8], [9]. Alternative architectures for mission critical communication are presented in [10]. Feasibility study on 3GPP mission critical multiparty communications is provided in [11]. The authors present realization of mission critical Push-to-Talk service and evaluate key performance indicators of the service. In [12], the authors propose an

Manuscript received 10 April, 2020; accepted 30 August, 2020.
The research was conducted under the Grant No. KP-06-H37/33 of project funded by Bulgarian National Science Fund, Ministry of Education and Science.

architecture for distributed mission critical Push-To-Talk service based on IP Multimedia Subsystem (IMS). In the proposed architecture, the application server is co-located with distributed core network functions for user traffic handling, while the control plane travels through the operator's centralized core network. In [13], the key performance indicators of mission critical Push-to-Talk service in 5G architecture are studied.

Our proposal for deployment of mission critical multiparty communication services is based on MEC and distributed dedicated core network functionality, including access and mobility management, session management, user data registry, and user plane functions. The benefit is that the signalling path does not need to traverse through the operator's core network.

The research novelty is in delegating the multiparty call control to mobile edge applications deployed in the vicinity of end users. The proposed open access to programmability of multiparty communications does not require deployment of IMS, and thus it reduces the costs and provides the necessary performance and reliability.

The rest of the paper is organized as follows. Next section describes how the proposed functionality can be deployed in 5G for mission critical and machine type communications. Section III provides overall description of the new API for multiparty multimedia communications by typical use cases. Section IV introduces how the API may be used by MEC applications and the information that can be exchanged. Section V illustrates the API feasibility by modelling the multiparty call state from network and application points of view. Some performance metrics of the proposed APIs are discussed in Section VI, where the injected latency is evaluated by emulation. The conclusion summarizes the novelty and benefits of the proposed API.

II. Deployment of API for Multiparty Communication Control at the Network Edge

Distributed core network functionality for mission critical and machine type communications can be deployed by virtualization of core network functions and customized to the specific requirements using the technique of network slicing [14], [15]. Customized core network functions, such as Access and mobility Management Function (AMF), Session Management Function (SMF), User Data Repository (UDR), User Plane Function (UPF), and Policy Control Function (PCF), run on an NFV platform. MEC applications for mission critical and machine type communications, which are run as virtual machines, as well as the mobile edge platform, which provides mobile edge services, can share the same NFV platform.

The deployment scenario for MEC and distributed virtualized core network functions are shown in Fig. 1.

We propose a new mobile edge service called "Multiparty Communication Control" (MPCC). The service provides open access to multiparty call control for MEC applications. The call control functions are part of the core network and may be accessed through the Network Exposure Function (NEF). The NEF securely exposes core network functionality and information provided by the network to the MEC platform and applications.

5G core network architecture is centred around services. The proposed MPCC service must use the NEF Nnef_AFsessionWithQoS service to create an Application Function (AF) session with required QoS [16], [19].

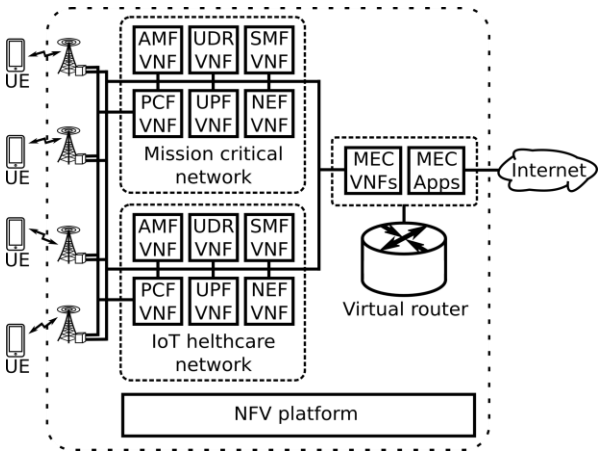

Fig. 1. MEC co-location with distributed virtualized core functionality.

The communication approach between core network services, mobile edge services, and applications follows REpresentational State Transfer (REST) architectural style. In RESTful API, each entity is represented as a uniquely identified resource with associated data, set of methods, and relationship to other resources. The RESTful communication follows request/response pattern and the resource methods correspond to the HTTP methods POST, GET, PUT, PATCH, and DELETE.

So, following the adopted architectural style, the proposed MPCC interfaces described in the next section are REST-based.

III. Overall Description of API for Multiparty Communication Control

This section describes the proposed API for multiparty multimedia call control. To clarify how the respective API functions can be provided, the corresponding core network services are also commented.

The 5G specifications provide access to monitoring of call related events and enable setting of application server sessions with specific quality of service, but do not provide functionality for external applications to manage multiparty calls. The proposed MPCC service enables mobile edge applications to create multimedia multiparty sessions and to manage dynamically the session participation. Using the MPCC interfaces, a mobile edge application may:

– Setup a multiparty call with specified quality of service;
– Add or remove a participant to or from the call;
– Retrieve information about multiparty call and call participants.

To setup a multiparty communication, a mobile edge application first creates a multimedia session in specific context that represents a virtual meeting. A unique session identifier is assigned and initially no participants are connected.

The mobile edge application subsequently may add participants to the multimedia session specifying the media

types. This results in setup of AF session with required QoS, PCF-initiated session management policy association and network triggered service request to the participant User Equipment (UE) in the network [17], [18], [19]. The multimedia session becomes active with first participant connecting. The mobile edge application can check the status of the session participants.

The mobile edge application may dynamically manage media streams for a multiparty session participant, e.g., adding a video component to a participant using only audio so far in the multiparty session. On application request for adding or removing a media stream, the network executes modification of AF session with QoS.

During the multiparty multimedia session, the mobile edge application may disconnect a participant from the session, or it may retrieve information about the multiparty session status and session participant status.

The end of the multimedia session occurs due to mobile application-initiated session termination or after all the participants have left the session. If the mobile edge application has specified the maximum session duration during session setup, the session ends on expiry of session duration. When the multiparty multimedia session maximum duration expires or the mobile edge application terminates the session, a removal of AF session with QoS takes place in the network [19].

The mission critical multiparty communications and health multiparty communications share a common base of functionalities which can be depicted by the following use cases.

A video analytic application detects increased anxiety in an elderly patient and initiates a multiparty communication that joins the patient, his or her physician, and a specialist to assess the patient's momentary condition and to provide patient with specific recommendations and counselling.

Another exemplary use case of healthcare communication is an analytic application which receives data about patient's treatment and status from an ambulance team during their way to the hospital, and meanwhile it makes a multiparty session with doctor's group at the emergency department to provide quick rescue to the patient.

To illustrate the proposed functionality, Fig. 2 shows the flow of initiation of multiparty multimedia session by a mobile edge application for the presented use case.

The MPCC service is in a role of AF. The sequence of steps in creating a multimedia multiparty session is as follows:

1. The mobile edge application using the MPCC interfaces requests a multiparty multimedia session establishment without any participants initially. The application provides a session description and maximum number of session participants. Optionally, the maximum duration of the multiparty session and the address of the multiparty session owner may also be provided.
2. The MPCC service stores the multiparty session data and responds to the application. The response includes the identifier of the created multiparty multimedia session.
3. The mobile edge application wishes to involve a participant in the multiparty multimedia session. To do this, the application sends a request for adding the participant, including the media types that are allowed for this participant. The request includes the multiparty session identifier and participant address.

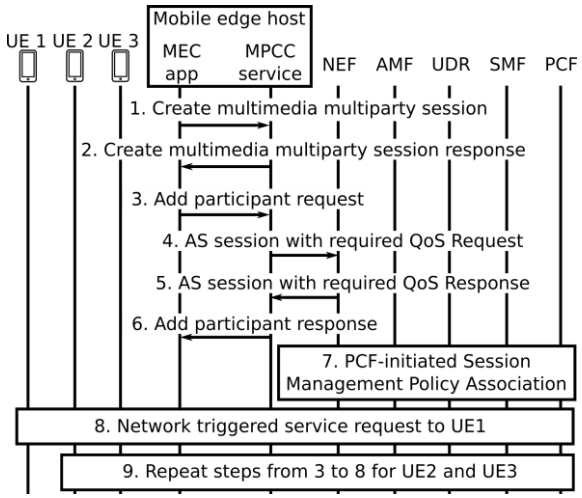

Fig. 2. A mobile edge application-initiated multiparty session.

4. The MPCC service invokes Nnef_AFsessionWithQoS_Create service operation which requests the network to setup an AF session with required QoS.
5. The NEF authorizes the requests, forwards it in the network, subscribes for UE reachability events, and responds to the MPCC service.
6. The MPCC service responds to the application request for adding a participant to the multiparty session.

7–8. The NEF triggers PCF initiated session management policy association and network triggered service request to the UE of the first multiparty session participant. The NEF is notified about UE reachability events, and in turn it notifies the MPCC service about the participant. The mobile edge application can query about multiparty session participant status.

9. When the mobile edge application wishes to invite the next participants in the multiparty session, the steps from 3 to 8 are repeated.

IV. Data Model and API Definition for Multiparty Communication Control

As a part of MPCC service specification, we build the data model and provide interface definition. This section describes the required information and message flows. It also provides detailed description on all information elements used for multiparty call control.

The entities multiparty session and session participants are all presented as RESTful resources. Figure 3 shows the MPCC resources organized in a tree structure.

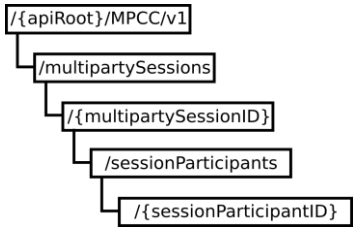

Fig. 3. Resource structure supported by the MPCC service.

Each resource is uniquely identified by its Uniform Resource Identifier (URI), and the common root of service resource may be published and discovered by service directory.

A. Multiparty Session Management

The *multipartySessions* resource represents all multiparty multimedia sessions created by mobile edge applications. The resource supports HTTP GET method, which retrieves a list of all active application-created multiparty sessions, and HTTP POST method, which creates a new multiparty multimedia session.

To initiate a new multiparty multimedia session, a mobile edge application sends a HTTP POST request to the *multipartySessions* resource with message body containing *multipartySessionData* data structure. The *multipartySessionData* data type is a JSON structure, where the attributes are given in Table I.

TABLE I. ATTRIBUTES OF MULTIPARTYSESSIONDATA DATA TYPE.

Attribute name	Type	Cardinality	Meaning
>timeStamp	TimeStamp	0…1	TimeStamp
>sessionDescription	String	0…1	Textual description of the multiparty session
>maxParticipant-Number	Integer	1	The maximal number of participants allowed
>maxDuration	Integer	1	Maximal session duration
>appInsID	String	1	The unique application instance identifier
>requestID	String	1	The request identifier allocated by the application
>chargingInfo	String	0…1	The tariff the session will be charged

The MPCC service creates a *multipartySessionID* resource representing the requested multiparty session and responds with "201 Created" with message body containing the *multipartySessionData* data structure and the allocated multiparty session identifier.

Figure 4 shows the flow of mobile edge application requesting creation of a multiparty multimedia session.

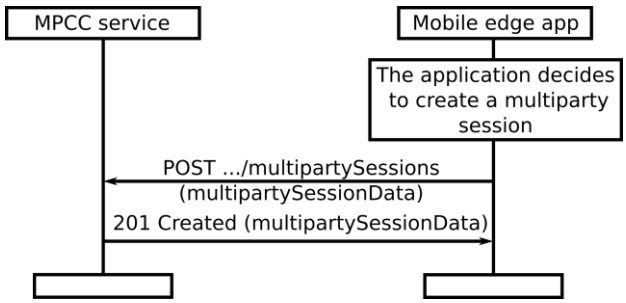

Fig. 4. Message flow of application-initiated multiparty multimedia session.

The *multipartySessionID* resource represents an existing multiparty multimedia session created by a mobile edge application. The resource supports HTTP method GET, which retrieves information about the multiparty session, HTTP methods PUT or PATCH, which update the multiparty multimedia session, and HTTP method DELETE, which requests termination of the multiparty multimedia session.

The mobile edge application can request the current status of a multiparty multimedia session. To do this, the application sends a HTTP GET request to the resource representing the respective multiparty multimedia session. The MPCC service responds with "200 OK" message containing the *multipartySessionInfo* data structure. The *multipartySessionInfo* data type is a JSON structure, where the attributes are given in Table II.

TABLE II. ATTRIBUTES OF MULTIPARTYSESSIONINFO DATA TYPE.

Attribute name	Type	Cardinality	Meaning
>timeStamp	TimeStamp	0…1	TimeStamp
>startTime	TimeStamp	1	The time the session begins
>activeTime	TimeStamp	1	The time at which the multiparty session was active
>sessionOwner	String	0…1	The owner of the session
>participantNumber	Integer	1…n	The current number of participants involved in the session
>maxParticipantNumber	Integer	1…n	The maximal number of participants allowed
>appInsID	String	1	The unique application instance identifier
>requestID	String	1	The request identifier allocated by the application
>sessionDescription	String	0…1	Textual description of the multiparty session

Figure 5 shows the flow of retrieving information about multiparty multimedia session.

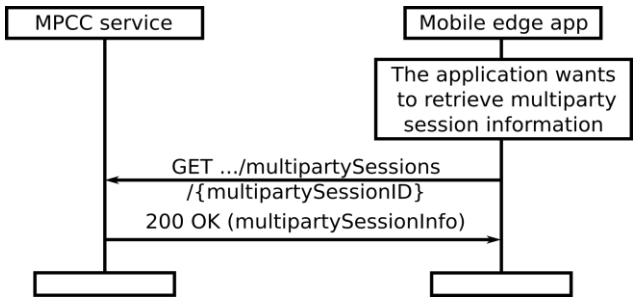

Fig. 5. Message flow of application-initiated multiparty multimedia session.

The mobile edge application can modify the multiparty multimedia session data partially (e.g., the maximum number of participants involved in the multiparty session or maximum session duration). To do this, the application sends an HTTP PATCH request to the resource representing the multiparty multimedia session with message body containing the modified *multipartySessionData* structure. The MPCC service modifies the attributes of the multiparty session and responds with "200 OK", including the modified multiparty session data.

Figure 6 shows the flow of modifying multiparty multimedia session data.

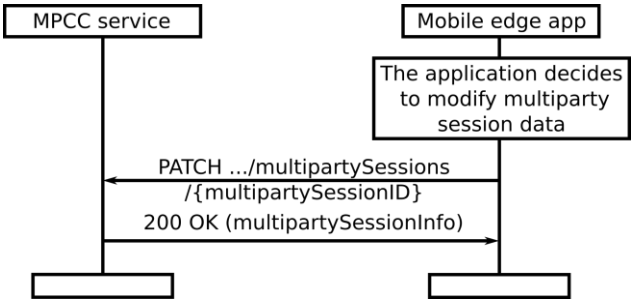

Fig. 6. Message flow of modifying multiparty multimedia session data.

When the mobile edge application decides to terminate a multiparty multimedia session, it sends an HTTP DELETE request to the resource representing the session. On receiving the request, the MPCC service deletes the respective resource and responds with "204 No Content". If the multiparty session status is active, the MPCC service invokes Nnef_AFWithQoS_Create service operation of NEF indicating the release of the AF session with QoS.

Figure 7 shows the flow of terminating a multiparty multimedia session.

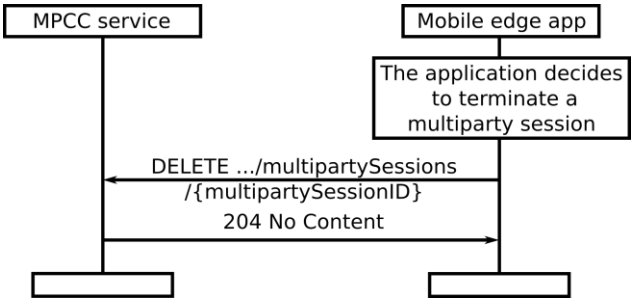

Fig. 7. Message flow of terminating a multiparty multimedia session.

B. Multiparty Session Participant Management

The *multipartySessionParticipants* resource represents all participants connected to the multiparty multimedia session. The resource supports HTTP POST and GET methods. A mobile edge application uses the GET method to retrieve a list of all participants involved in the multiparty session. The MPCC responds with "200 OK" message, including the list of participants URIs.

The application can add a participant to the multiparty multimedia session. To do so, the application sends an HTTP POST requests to the *sessionParticipants* resource of the respective multiparty session. The message body contains *participantData* data structure which specifies the participant URI and the information about media streams used for the initial connection. Each media stream is defined by its type, QoS Class Identifier, and priority level. The *participantData* data type is a JSON structure, where the attributes are given in Table III.

Upon receiving the request, the MPCC service creates a *sessionParticipant* resource representing the participant and responds with "201 Created" message, including the data of the participant. The MPCC invokes the Nnef_AFWithQoS_Create service operation of NEF requesting the network to set up an AF session with specified QoS with implicit subscription for bearer level events. The NEF authorizes the request and forwards it into the network. When the participant connects to the session, the NEF invokes Nnef_AFWithQoS_Notify service operation to notify the MPCC service about the bearer level event as described in [19].

TABLE III. ATTRIBUTES OF PARTICIPANTDATA DATA TYPE.

Attribute name	Type	Cardinality	Meaning
>timeStamp	TimeStamp	0...1	TimeStamp
>participantInfo	Structure	1…n	The initial information about the participant
>>participantURI	URI	1	The participant address
>>mediaInfo	Structure	1…n	The media information
>>>media	String	1	The media type (e.g., voice, video, data, text)
>>>QCI	Integer	1	The QoS Class identifier as defined by 3GPP
>>>priorityLevel	Integer	1	Allocation and Retention Priority as defined by 3GPP
>requestID	String	1	The request identifier allocated by the application

Figure 8 shows the flow of adding a participant to a multiparty multimedia session.

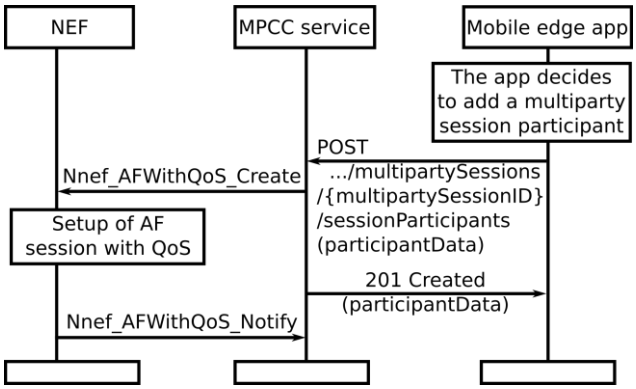

Fig. 8. Flow of adding a participant to a multiparty multimedia session.

The *sessionParticipantID* resource represents an existing participant involved in a multiparty multimedia session. It supports HTTP methods GET, PUT, and DELETE.

The mobile edge application can retrieve information about a multiparty session participant by sending an HTTP GET request to the respective *sessionParticipantID*

resource. The MPCC service responds with "200 OK" message which transfers the *participantStatus* data type. The *participantStatus* data type is JSON structure, and its attributes are given in Table IV.

Figure 9 shows the flow of retrieving information about multiparty session participant.

The mobile edge application can add or remove media for participant involved in a multiparty multimedia session. To do this, the application sends an HTTP PATCH request to the *sessionParticipantID* resource containing in its body the updated media information. The MPCC invokes the Nnef_AFWithQoS_Update service operation of NEF requesting the network to update some of the properties of the established AF session with specified QoS.

TABLE IV. ATTRIBUTES OF PARTICIPANTINFO DATA TYPE.

Attribute name	Type	Cardinality	Meaning
>timeStamp	TimeStamp	0…1	TimeStamp
>participantInfo	Structure	1…n	The initial information about the participant
>>participantURI	URI	1	The participant address
>>mediaInfo	Structure	1…n	The media information
>>>media	String	1	The media type (e.g., voice, video, data, text)
>>>QCI	Integer	1	The QoS Class identifier as defined by 3GPP
>>>PriorityLevel	Integer	1	Allocation and Retention Priority as defined by 3GPP
currentStatus	Enumerated	1	The current participant status: 1 = connected; 2 = disconnected

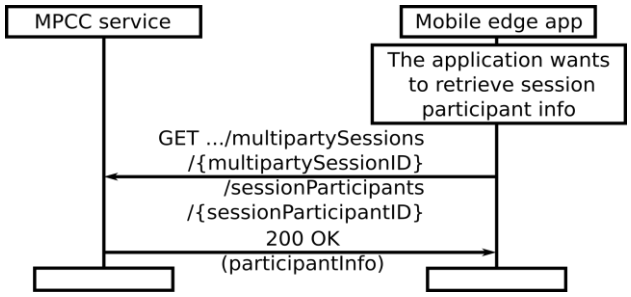

Fig. 9. Flow of retrieving information about multiparty session participant.

The NEF authorizes the request and forwards it to the network. When the established AF session with specified QoS is updated, the NEF invokes *Nnef_AFWithQoS_Notify* service operation to notify the MPCC service about the bearer level event as described in [19]. The MPCC service responds to the application with "200 OK" message with body containing updated participant media information. Figure 10 shows the flow of information update about participant connected to a multiparty multimedia session.

When the mobile edge application decides to remove a participant from a multiparty multimedia session, it sends an HTTP DELETE request to the *sessionParticipantID* resource containing in its body the updated media information. The MPCC invokes the Nnef_AFWithQoS_Delete service operation of NEF requesting the network to remove all properties of the established AF session with specified QoS. The NEF authorizes the request and interacts with PCF to terminate the session. The MPCC service responds to the application with "204 No Content" message.

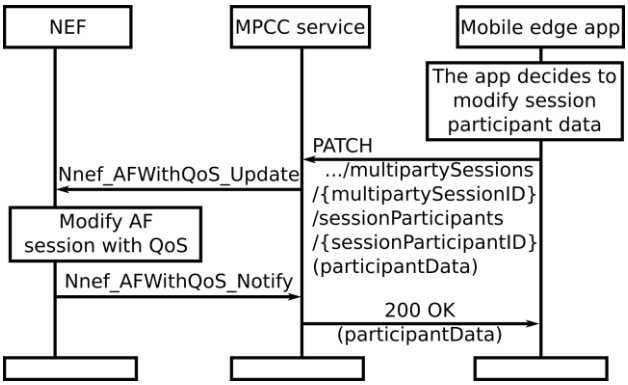

Fig. 10. Flow of multiparty session participant information update.

Figure 11 shows the flow of removing a participant from a multiparty multimedia session.

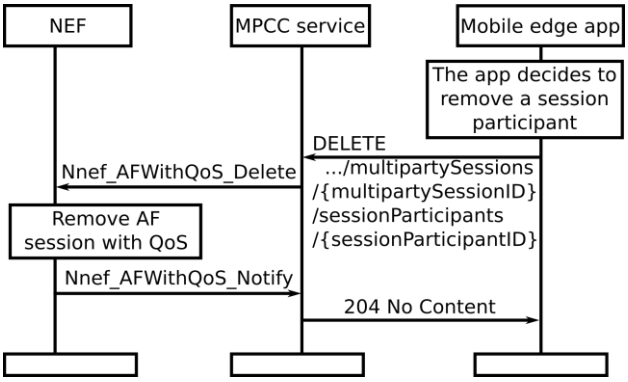

Fig. 11. Flow of removing a participant from a multiparty multimedia session.

Table V summarizes the API resources and supported methods for multiparty communication control.

TABLE V. API RESOURCES AND SUPPORTED METHODS FOR MULTIPARTY COMMUNICATION CONTROL.

Resource name	Resource URI	HTTP method	Meaning
All multiparty multimedia sessions initiated by application	/multipartySessions	GET POST	Retrieves the list of all multiparty sessions Creates a new multiparty session
An existing multiparty multimedia session	/multipartySessions /{multipartySessionID}	GET PATCH DELETE	Retrieves information about existing multiparty session Modifies existing multiparty session Cancels existing multiparty session
All participants of a multiparty multimedia session	/multipartySessions /{multipartySessionID} /sessionParticipants	GET POST	Retrieves list of all participants in a multiparty session Adds a new participant to a multiparty session

Resource name	Resource URI	HTTP method	Meaning
Existing participant of a multiparty multimedia session	/multipartySessions /{multipartySessionID} /sessionParticipants /{sessionParticipantID}	GET	Retrieves information about existing participant in a multiparty session
		PATCH	Updates data of a participant in a multiparty session
		DELETE	Removes a participant from a multiparty session

V. FEASIBILITY STUDY

To assess the feasibility of the proposed API, we model the visions of the network and mobile edge application on the state of a multiparty multimedia session. The network and mobile edge application must have synchronized visions and the multiparty multimedia session state models must expose equivalent behaviour.

In this section, models representing the multiparty multimedia session state are proposed and described formally using the concept of Labelled Transition Systems (LTS). A mathematical proof for behavioural equivalence of the models is provided formally using the concepts of bi-simulation.

Figure 12 shows a simplified multiparty session state model supported by application.

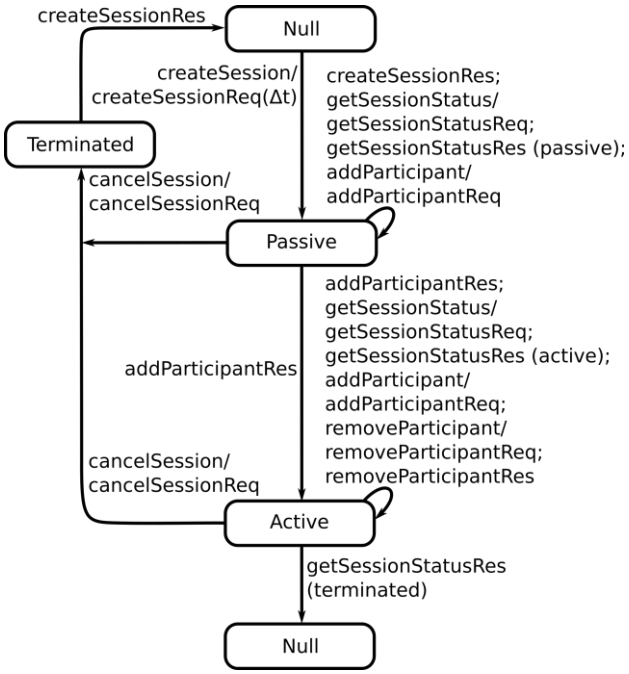

Fig. 12. A simplified multiparty session state model supported by application.

For simplicity sake, the model does not show application's queries about multiparty participant status.

The Null state is the initial one, where the multiparty multimedia session does not exist. In Passive state, the multiparty multimedia session is created with maximum duration Δt, and no participants are connected yet. When the application adds a multiparty participant, the state moves to Active.

In Active state, the multiparty multimedia session has participant(s), and the application may request adding or removing a multiparty participant. When the application decides to cancel the multiparty session, the state becomes Terminated state. In each state, the application may query the MPCC service about the multiparty session status.

An LTS is formal representation of a state model defined by a set of states, a set of inputs, a set of transitions, and a set of initial states.

By T_{app}, it is denoted an LTS representing the multiparty session state model supported by a mobile edge application. $T_{app} = (S_{app}, Inp_{app}, Trans_{app}, s^0_{app})$, where:

S_{app} = {Null [s^{app}_1], Passive[s^{app}_2], Active[s^{app}_3], Terminated [s^{app}_4];

Inp_{app} = {createSession [t^{app}_1], createSessionRes [t^{app}_2], getSessionStatus [t^{app}_3], getSessionStatusRes(passive) [t^{app}_4], addParticipant [t^{app}_5], addParticipantRes [t^{app}_6], getSessionStatusRes(active) [t^{app}_7], removeParticipant [t^{app}_8], removeParticipantRes [t^{app}_9], getSessionStatusRes(terminated) [t^{app}_{10}], cancelSession [t^{app}_{11}], cancelSessionRes [t^{app}_{12}]};

$Trans_{app}$ = {(s^{app}_1, t^{app}_1, s^{app}_2), (s^{app}_2, t^{app}_2, s^{app}_2), (s^{app}_2, t^{app}_3, s^{app}_3), (s^{app}_2, t^{app}_4, s^{app}_2), (s^{app}_2, t^{app}_5, s^{app}_2), (s^{app}_2, t^{app}_6, s^{app}_3), (s^{app}_3, t^{app}_6, s^{app}_3), (s^{app}_3, t^{app}_3, s^{app}_3), (s^{app}_3, t^{app}_7, s^{app}_3), (s^{app}_3, t^{app}_5, s^{app}_3), (s^{app}_3, t^{app}_8, s^{app}_3), (s^{app}_3, t^{app}_9, s^{app}_3), (s^{app}_3, t^{app}_{10}, s^{app}_1), (s^{app}_3, t^{app}_{11}, s^{app}_4), (s^{app}_2, t^{app}_{11}, s^{app}_4), (s^{app}_4, t^{app}_{12}, s^{app}_1)};

$s^0_{app} = s^{app}_1$.

In the formal model description, short notations of state and input names are given in brackets.

Figure 13 shows the multiparty state model supported by the network.

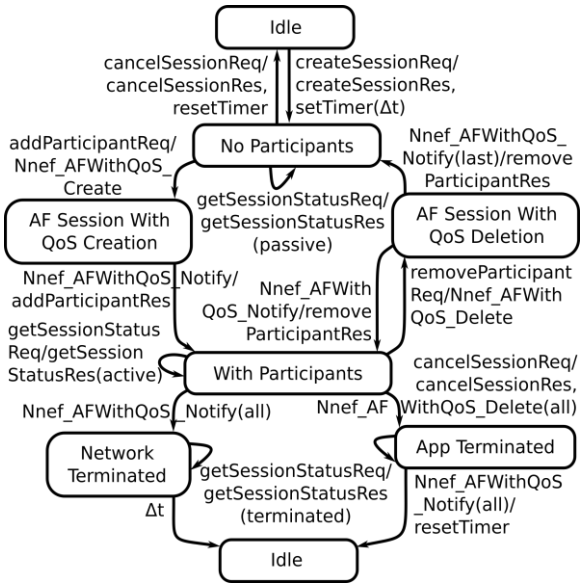

Fig. 13. A simplified multiparty session state model supported by the network.

In Idle state, there is no multiparty multimedia session. When the MPCC service receives a request for multiparty session creation, it creates an object representing the session with maximal duration Δt, and the session state becomes NoParticipants. In NoParticipants state, the multiparty multimedia session is created with no participants.

Upon receiving a request for adding a participant, the MPCC service initiates a procedure for establishment of AF session with required QoS to the UE. In

AFSessionWithQoSCreation state, the MPCC service waits for notification about UE reachability. The multiparty session becomes WithParticipants when at list one UE is connected. In WithParticipants state, a request for adding a participant may be received. Upon receiving a request for removing a participant, the MPCC service initiates a procedure for release of the AF session with the UE. In AFSessionWithQoSDeletion state, the MPCC service waits for notification about UE reachability. Upon receiving a request for the multiparty session termination, the MPCC service initiates a procedure to terminate the multiparty session and the multiparty session state becomes AppTerminated. In AppTerminated state, the MPCC service waits for notification about session termination. In NetworkTerminated state, the multiparty session is terminated by the network, and the MPCC service maintains the multiparty session state until the maximum session state expires and moves to Idle state.

In each state, when an application request for multiparty session state status is received, the MPCC service sends a response.

By T_{mec}, it is denoted an LTS representing the multiparty multimedia session state model supported by the network $T_{mec} = (S_{mec}, Inp_{mec}, Trans_{mec}, s^0_{mec})$, where:

S_{mec} = {Idle [s^{mec}_1], NoParticipants [s^{mec}_2], AFSessionWithQoSCreation [s^{mec}_3], WithParticipants [s^{mec}_4], AFSessionWithQoSDeletion [s^{mec}_5], AppTerminated [s^{mec}_6], NetworkTerminated [s^{mec}_7]};

Inp_{mec} = {createSessionReq(Δt) ([t^{mec}_1], getSessionStatusReq [t^{mec}_2], addParticipantReq [t^{mec}_3], Nnef_AFWithQoS_Notify [t^{mec}_4], removeParticipantReq [t^{mec}_5], Nnef_AFWithQoS_Notify(all) [t^{mec}_6], Δt [t^{mec}_7], cancelSessionReq [t^{mec}_8], Nnef_AFWithQoS_Notify(last) [t^{mec}_4]};

$Trans_{mec}$ = {(s^{mec}_1, t^{mec}_1, s^{mec}_2), (s^{mec}_2, t^{mec}_2, s^{mec}_2), (s^{mec}_2, t^{mec}_3, s^{mec}_3), (s^{mec}_3, t^{mec}_4, s^{mec}_4), (s^{mec}_4, t^{mec}_5, s^{mec}_5), (s^{mec}_5, t^{mec}_4, s^{mec}_4), (s^{mec}_4, t^{mec}_8, s^{mec}_6), (s^{mec}_4, t^{mec}_2, s^{mec}_4), (s^{mec}_6, t^{mec}_2, s^{mec}_6), (s^{mec}_6, t^{mec}_6, s^{mec}_1), (s^{mec}_4, t^{mec}_6, s^{mec}_7), (s^{mec}_7, t^{mec}_2, s^{mec}_7), (s^{mec}_7, t^{mec}_7, s^{mec}_1), (s^{mec}_2, t^{mec}_8, s^{mec}_1), (s^{mec}_5, t^{mec}_9, s^{mec}_2)};

$s^0_{mec} = s^{mec}_1$.

Both models may be regarded as concurrent processes which external actions are identical, i.e., visible process behaviours are equivalent. To prove that the LTS behaviours are equal, we use bi-simulation concept. The bi-simulation concept enables to study the behavioural features of the processes and to abstract from their details. Bi-simulation is considered as one of the most important mathematical tools in concurrency theory of computer science.

The formal model validation enables to prove that both LTSs behave the same way, i.e., the network view on multiparty multimedia session state is synchronized with the mobile edge application's view on the multiparty multimedia session state. In strong bi-simulation, there must be a strong relationship between each transition in the one LTS and the respective transition in the other LTS, i.e., both LTSs need to display the same result. In weak bi-simulation, there may exist internal transitions that can be discarded.

Proposition: Both LTSs T_{app} and T_{mec} have a weak bi-simulation relationship.

Proof: The weak bi-simulation relationship requires identification of pairs of LTSs states that match each other's transitions. Let $R_{app\&mec}$ = {(s^{app}_1, s^{mec}_1), (s^{app}_2, s^{mec}_2), (s^{app}_3, s^{mec}_4)}. Then, the following functional mapping between the transitions in T_{app} and T_{mec} exists:

1. The mobile edge application requests a multiparty multimedia session creation and the MPCC creates an object representing the session: for (s^{app}_1, t^{app}_1, s^{app}_2), (s^{app}_2, t^{app}_2, s^{app}_2) ∃ (s^{mec}_1, t^{mec}_1, s^{mec}_2).
2. While the multiparty session is passive with no participants connected, the application queries about the multiparty session state: for (s^{app}_2, t^{app}_3, s^{app}_3), (s^{app}_2, t^{app}_4, s^{app}_2) ∃ (s^{mec}_2, t^{mec}_2, s^{mec}_2).
3. The mobile edge application invites the first session participant and the MPCC service initiates an establishment of a session with his/her UE, and the multiparty session becomes active: for (s^{app}_2, t^{app}_5, s^{app}_2), (s^{app}_2, t^{app}_6, s^{app}_3) ∃ (s^{mec}_2, t^{mec}_3, s^{mec}_3), (s^{mec}_3, t^{mec}_4, s^{mec}_4).
4. While the multiparty session is active, the application queries about multiparty session state: for (s^{app}_3, t^{app}_3, s^{app}_3), (s^{app}_3, t^{app}_7, s^{app}_3) ∃ (s^{mec}_4, t^{mec}_2, s^{mec}_4).
5. The mobile edge application removes a participant (not the last one) from the multiparty session, and the MPCC service initiates a deletion of the AF session with his/her UE, and the multiparty session states remains active: for (s^{app}_3, t^{app}_8, s^{app}_3), (s^{app}_3, t^{app}_9, s^{app}_3) ∃ (s^{mec}_4, t^{mec}_5, s^{mec}_5), (s^{mec}_5, t^{mec}_4, s^{mec}_4).
6. While the multiparty session is active, the application requests multiparty session cancelation, and the MPCC service initiates release of all session participants: for (s^{app}_3, t^{app}_{11}, s^{app}_4), (s^{app}_4, t^{app}_{12}, s^{app}_1) ∃ (s^{mec}_4, t^{mec}_8, s^{mec}_6), (s^{mec}_6, t^{mec}_2, s^{mec}_6), (s^{mec}_6, t^{mec}_6, s^{mec}_1).
7. While the multiparty session is passive, the application requests multiparty session cancelation, and the MPCC service deletes the object representing the multiparty session: for (s^{app}_2, t^{app}_{11}, s^{app}_4), (s^{app}_4, t^{app}_{12}, s^{app}_1) ∃ (s^{mec}_4, t^{mec}_8, s^{mec}_6), (s^{mec}_6, t^{mec}_2, s^{mec}_6), (s^{mec}_6, t^{mec}_6, s^{mec}_1).
8. The multiparty session ends in the network and when the application queries about its state, the MPCC service responds that the session is terminated: for (s^{app}_3, t^{app}_{10}, s^{app}_1) ∃ (s^{mec}_4, t^{mec}_6, s^{mec}_7), (s^{mec}_7, t^{mec}_2, s^{mec}_7), (s^{mec}_7, t^{mec}_7, s^{mec}_1).
9. The application removes the last participant and terminates the multiparty multimedia session: for (s^{app}_3, t^{app}_8, s^{app}_3), (s^{app}_3, t^{app}_{11}, s^{app}_4), (s^{app}_4, t^{app}_{12}, s^{app}_1) ∃ (s^{mec}_4, t^{mec}_5, s^{mec}_5), (s^{mec}_5, t^{mec}_9, s^{mec}_2), (s^{mec}_2, t^{mec}_8, s^{mec}_1).

Therefore, T_{app}, and T_{mec} have a weak bi-simulation relationship, i.e., they expose equivalent behaviour. ■

The formal model validation is useful in API design phase, and during API realization it can be used to prove the compliance of realization with its specification.

VI. Assessment of API Performance

One of the Key Performance Indications of MEC is latency which should be defined on per service basis [20]. In this section, we evaluate theoretically the latency of the control plane injected by the proposed API. The latency is evaluated for API requests for session creation and for API requests for participant managing.

Round-Trip-Time (RTT) latency is measured as the time taken for an HTTP request generated from a mobile edge application to get to the UE to be processed, to be answered, and to get back to the destination. In case of MEC co-location with distributed core network functions, the time for communication between the mobile edge application, mobile edge service, and core network functions can be regarded as negligible. HTTP methods are processed by a mobile edge application, mobile edge service, and core network (CN).

Adding a participant in a multiparty multimedia session includes session set-up time which involves the times required for processing in CN, Radio Access Network (RAN), and UE, as well as the time required for message transfer over the interface between CN and RAN and the interface between UE and RAN.

An example HTTP request for multiparty session creation that uses the proposed API looks like the following:

```
POST http://example.com/MPCC/v1/multipartySessions
HTTP/1.1
Accept: application/json
Content-type: application/json
Content-length: 240

{
 "timeStamp": "Mon Jan 20 17:30:50 EET 2020",
 "sessionDescription": "Meeting",
 "maxParticipantsNumber": 3,
 "maxDuration": 360,
 "appInsID": "93462efb-d072-46a9-9d7b-512842949a47",
 "requestID": "041f9b0e-2f5b-472f-b473-9a5323e3b0b0"
}
```

The respective HTTP response of the request for multiparty session creation sent by the proposed MPCC service looks like the following:

```
HTTP/1.1 201
Location:
http://example.com/MPCC/v1/multipartySessions/
/c853b5c2-e903-40ad-ab67-2639eabaedbb
Content-type: application/json
Content-length: 240

{
 "timeStamp": "Mon Jan 20 17:30:50 EET 2020",
 "sessionDescription": "Meeting",
 "maxParticipantsNumber": 2,
 "maxDuration": 360,
 "appInsID": "93462efb-d072-46a9-9d7b-512842949a47",
 "requestID": "041f9b0e-2f5b-472f-b473-9a5323e3b0b0"
}
```

According to [21], [22], the time budget for local task execution can be calculated as

$$T_i = D_i \times X/f, \quad (1)$$

where D_i is the data size (in bits), X_i is the computational workload (in mobile edge server's CPU cycles per bit), and f is the frequency of the mobile edge server's CPU. The data size is the number of symbols in the request and response of MPCC API. The time budget for processing HTTP requests and responses at the mobile edge server depends on mobile edge server characteristics and the following values are used: the CPU frequency f of MEC server is set to 2.2 GHz and the computational workload X is 1200 cycles/bit. The data size of a mobile edge application request for multiparty session creation is 371 bytes (371×8 bits), and the data size of the MEC platform response of the request for multiparty session creation is 392 bytes (392×8 bits).

So, the time required for the example HTTP request for multiparty session creation composition at the mobile edge application and processing at the MEC platform is given as

$$T_{req}^{Session} = D_{req}^{app} \times X/f + D_{req}^{mec} \times X/f = 1{,}7808\ ms \quad (2)$$

The time required for the composition at the MEC platform and processing by the mobile edge application of the example HTTP response is given as

$$T_{res}^{Session} = D_{res}^{app} \times X/f + D_{res}^{mec} \times X/f = 1{,}8816\ ms \quad (3)$$

The time budget $T_{MPCC}^{Session}$ for application-initiated multiparty multimedia session creation introduced by the proposed API is $T_{MPCC}^{Session} = 3{,}6624\ ms$.

An examplary request for adding of a participant to a multiparty session, initiated by mobile edge application which uses the proposed API, looks like the following:

```
POST http://example.com/MPCC/v1/multipartySessions/
/c853b5c2-e903-40ad-ab67-
2639eabaedbb/sessionParticipants HTTP/1.1
Accept: application/json
Content-type: application/json
Content-length: 359

{
 "timeStamp": "Mon Jan 20 17:30:50 EET 2020",
 "participantInfo": {
  "participantURI": "111218115@example.com",
  "mediaInfo": [
   {"media": "voice", "QCI": 65, "priorityLevel": 0.7},
   {"media": "video", "QCI": 67, "priorityLevel": 1.5}
  ]
 },
 "appInsID": "93462efb-d072-46a9-9d7b-512842949a47",
 "requestID": "905ab60e-8e2d-4ab2-a3a4-da698866475f"
}
```

The corresponding HTTP response is as follows:

```
HTTP/1.1 201
Location:
http://example.com/MPCC/v1/multipartySessions/
/c853b5c2-e903-40ad-ab67-
2639eabaedbb/sessionParticipants/1cdbf952-1dac-48d1-
b668-0f3ee63f8361
Content-type: application/json
Content-length: 359

{
 "timeStamp": "Mon Jan 20 17:30:50 EET 2020",
 "participantInfo": {
  "participantURI": "111218115@example.com",
  "mediaInfo": [
   {"media": "voice", "QCI": 65, "priorityLevel": 0.7},
```

```
  {"media": "video", "QCI": 67, "priorityLevel": 1.5}
 ]
},
"appInsID": "93462efb-d072-46a9-9d7b-512842949a47",
"requestID": "905ab60e-8e2d-4ab2-a3a4-da698866475f"
}
```


The time budget $T_{MPCC}^{Participant}$ for adding a participant to a multiparty multimedia session introduced by the proposed API is $T_{MPCC}^{Participant}$ = 5.3520 *ms*.

In addition, to theoretically evaluated latency, an experiment for API functionality is conducted. The experiment includes pushing a batch of one hundred thousand operations (POST requests for multiparty session creation and the respective "200 OK" responses). The configuration of the experiment includes client and service side with 1 GbE between them. The client is run on a host equipped with Intel processor of 7th generation, running at 3.4 GHz, with 8 cores and 8 GB of RAM. The server is running on Intel same generation at 2.6 GHz, with 6 cores and 16 GB of RAM. The client is implemented in Java and the server is implemented using Vert.x, which allows REST-based interface to be exposed toward the client, and Redis, which serves as in-memory store configured to work in single node mode, i.e., without any clustering.

In Fig. 14, it is depicted a decimated portion of operations latencies where the initial part is clearly different with respect of the following, and the effect is based on the so-called "warm-up phase". Should this effect have to be taken into consideration when a Service Level Agreement (SLA) is prepared, then it must be described separately, but the rest of the results presented are about the part of so called "steady state".

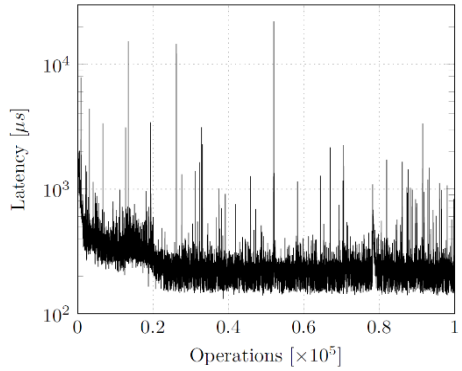

Fig. 14. Record of latencies for a sequence of 10^5 operations with service "warm-up phase" and "steady state" service phase.

In Fig. 15, it is depicted the "steady state" of latency as an ingredient of a SLA and the focus is set on the worst-case part as a limiting factor for the latency "budget" of the service. About 99.5 % of all requests have under millisecond latency, but part of the rest may hit even 30 *ms*. This is specific for both cases of the experiment - with a single serving instance at the endpoint and with 4 instances.

In Fig. 16, it is depicted the half-millisecond part of the latency "population" in steady state as it constitutes over 95 % of all recorded POST trials. By increasing the number of instances from 1 to 4, it becomes clear, that while keeping almost the same shape, the probability density function of latency moves to lower values as whole.

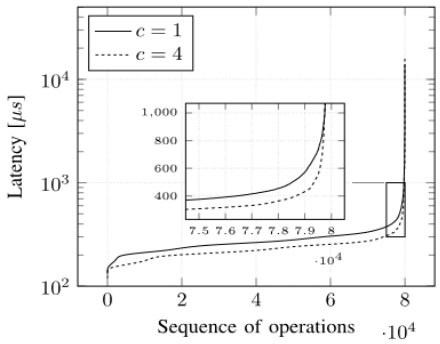

Fig. 15. SLA perspective of latencies as KPI ("steady state" case): c = 1 - single instance of serving RESTful endpoint; c = 4 - four instances.

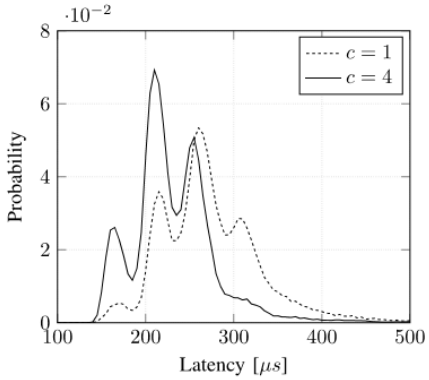

Fig. 16. Probability density functions as profile for the service latency in "steady state" phase (legend as previous).

In Fig. 17, it is depicted the Gaussian Mixture Model (GMM) of the "steady state" latency in the case of 4 serving instances. Here, the aim of the modelling part has at least three purposes: 1) to reduce the amount of data stored for simulation and development purposes, i.e., it is much more practical to generate expected latency from a 5-component model rather than keeping in cash 100000 samples; 2) to help in SLA formulation for the latency as KPI; 3) to help KPI monitoring for eventual slow shifts toward higher values of latency. The GMM parameters are given in Table VI.

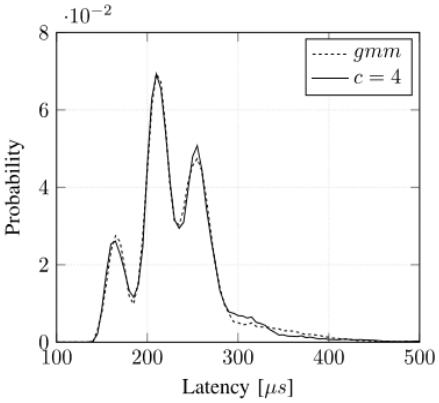

Fig. 17. 5-component GMM of four-instances service latency profile compared to raw data density in "steady state" phase.

TABLE VI. LATENCY GMM PARAMETERS.

Probability	Mean	Variance
0.131867179	163.2278	90.18257
0.281793207	208.3387	126.5642
0.323640009	251.4111	223.7910
0.155606968	289.7284	4445.668
0.007083637	1646.5074	11803980.7

The evaluated latency of the proposed mobile edge API for multiparty communication control shows that the network programmability at the network edge might enable low latency applications.

VII. CONCLUSIONS

The open access to multiparty call control at the network edge may be beneficial for mission critical scenarios and IoT healthcare use cases. It enables creation, modification, and termination of a multiparty multimedia session by applications adding and removing multiparty session participants dynamically, as well as management of media streams used. The open access to multiparty call control may be provided using MEC technology. In such deployments, MEC platform, which provides mobile edge services and applications, is co-located with distributed core functions.

The main paper contribution is the specification of API which enables applications to create and control sessions with specific QoS. The proposed API provides high level of abstraction hiding telecommunication details. The proposed functionality is illustrated by typical use cases. The API design includes specification of message flows and the required information. As the API design follows the REST architectural style, the service-relevant objects are represented as uniquely identified resources organized in a tree structure. The API defines the methods supported by the resources. As the mobile edge platform and the application must be synchronized in the service context, the feasibility study of the proposed API is illustrated by models representing the MEC platform's and application's views on the multiparty multimedia session state. The models are formally described and validated using the concept of bi-simulation. The latency introduced by the proposed API is assessed theoretically and by emulation.

The proposed API for multiparty call control enables dedicated third-party applications to react promptly on situations requiring management of multiparty sessions with specific QoS and saves the backhaul network resources.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

[1] Examples of software functions for which the FDA will exercise enforcement discretion, US Food and Drug Administration, Sept. 2019. [Online]. Available: https://www.fda.gov/medical-devices/device-software-functions-including-mobile-medical-applications/examples-software-functions-which-fda-will-exercise-enforcement-discretion

[2] N. A. Mohammed, A. M. Mansoor, and R. B. Ahmad, "Mission-critical machine-type communication: An overview and perspectives towards 5G", *IEEE Access*, vol. 7, pp. 127198–127216, 2019. DOI: 10.1109/ACCESS.2019.2894263.

[3] O. Balakci *et al.*, "A Flexible network architecture for 5G systems, wireless communications and mobile computing, *Wireless Communications and Mobile Computing*, vol. 2019, article ID 5264012, 2019. DOI: 10.1155/2019/ 5264012.

[4] N. Al-Quzweeni, A. Q. Lawey, T. E. H. Elgorashi, and J. M. H. Elmirghani, "Optimized energy aware 5G network function virtualization", *IEEE Access*, vol. 7, pp. 44939–44958, 2019. DOI: 10.1109/ACCESS.2019.2907798.

[5] T. Lin and Z. Zhou, "NFV-enabled network slicing", in *Proc. of IEEE International Conference on Communications Workshops (ICC Workshops)*, Kansas City, MO, 2018, pp. 1–6. DOI: 10.1109/ICCW.2018.8403497.

[6] E. Pencheva, I. Atanasov, and V. Vladislavov, "Mission critical messaging using multi-access edge computing", *Cybernetics and Information Technology*, vol. 19, no. 4, pp. 73–89, 2019. DOI: 10.2478/cait-2019-0037.

[7] Q.-V. Pham *et al.*, "A survey of multi-access edge computing in 5G and beyond: Fundamentals, technology integration, and state-of-the-art", *IEEE Access*, vol. 8, pp. 116974–117017, 2020. DOI: 10.1109/ACCESS.2020.3001277.

[8] 3GPP TS 22.179 Technical Specification Group Services and System Aspects, Mission Critical Push-to-Talk (MCPTT), Stage 1, Release 17, v17.0.0, 2019.

[9] 3GPP TS 23.179 Technical Specification Group Services and System Aspects, Functional architecture and information flows to support mission critical communication services, Stage 2, Release 13, v13.5.0, 2017.

[10] "Network 2020: Mission Critical Communications", White Paper, GSMA, 2020. [Online]. Available: https://www.gsma.com/futurenetworks/resources/network-2020-mission-critical-communications/

[11] S. W. Choi, Y. Song, W. Shin, and J. Kim, "A feasibility study on mission-critical push-to-talk: Standards and implementation perspectives", *IEEE Communications Magazine*, vol. 57, no. 2, pp. 81–87, Feb. 2019. DOI: 10.1109/MCOM.2018.1700886.

[12] R. Solozabal, A. Sanchoyerto, E. Atxutegi, B. Blanco, J. O. Fajardo, and F. Liberal, "Exploitation of mobile edge computing in 5G distributed mission-critical push-to-talk service deployment", *IEEE Access*, vol. 6, pp. 37665–37675, 2018. DOI: 10.1109/ACCESS.2018. 2849200.

[13] A. Sanchoyerto, R. Solozabal, B. Blanco, and F. Liberal, "Analysis of the impact of the evolution toward 5G architectures on mission critical push-to-talk services", *IEEE Access*, vol. 7, pp. 115052–115061, 2019. DOI: 10.1109/ACCESS.2019.2930936.

[14] S. Kekki *et al.*, "MEC in 5G Networks", ETSI White Paper, no. 28, 2018. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf

[15] N. Raveendran, Y. Zha, Y. Zhang, X. Liu, and Z. Han, "Virtual core network resource allocation in 5G systems using three-sided matching", in *Proc. of ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, Shanghai, China, 2019, pp. 1–6. DOI: 10.1109/ICC.2019.8762095.

[16] 3GPP TS 29.513 Technical Specification Group Core Network and Terminals, System Architecture for the 5G System (5GS), Stage 2, Release 16, v16.3.0, 2019.

[17] 3GPP TS 23.502 Technical Specification Group Services and System Aspects, Procedures for the 5G System (5GS), Stage 2, Release 16, v16.2.0, 2019.

[18] 3GPP TS 29.522 Technical Specification Group Core Network and Terminals, 5G System, Network Exposure Function Northbound APIs, Stage 3, Release 15, v15.2.0, 2018.

[19] 3GPP TS 29.122 Technical Specification Group Core Network and Terminals, T8 reference point for Northbound APIs, Release 16, v16.3.0, 2019.

[20] ETSI GS MEC-IEG 006 Mobile Edge Computing, Market Acceleration, MEC Metrics Best Practice and Guidelines, v1.1.1, 2017.

[21] K. Cheng, Y. Teng, W. Sun, A. Liu, and X. Wang, "Energy-efficient joint offloading and wireless resource allocation strategy in multi-MEC server systems", in *Proc. of IEEE International Conference on Communications (ICC)*, Kansas City, MO, 2018, pp. 1–6. DOI: 10.1109/ICC.2018.8422877.

[22] F. Wang, J. Xu, X. Wang, and S. Cui, "Joint offloading and computing optimization in wireless powered mobile-edge computing system", in *Proc. of IEEE International Conference on Communications (ICC)*, 2017, pp. 1–14. DOI: 10.1109/ICC.2017.7997477.