1-1-2016

An Architecture for QoS-Enabled Mobile Video Surveillance Applications in a 4G EPC and M2M Environment

Mohammad Abu-Lebdeh
Concordia University

Fatna Belqasmi
Zayed University

Roch Glitho
Concordia University

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 10, 2016, accepted July 6, 2016, date of publication July 21, 2016, date of current version August 26, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2592919

An Architecture for QoS-Enabled Mobile Video Surveillance Applications in a 4G EPC and M2M Environment

MOHAMMAD ABU-LEBDEH[1], FATNA BELQASMI[2], AND ROCH GLITHO[1,3]
[1]Concordia University, Montréal, H3G 1M8, Canada
[2]Zayed University, Abu Dhabi 144534, United Arab Emirates
[3]University of Western Cape, Bellville 7535, South Africa

Corresponding author: M. Abu-Lebdeh (mohammad.abulebdeh@gmail.com)

This work was supported in part by the Discovery Grant through the Canadian National Science and Engineering Council and in part by the Canada Research Chair in End User Service Engineering for Communications Networks.

ABSTRACT Mobile video surveillance applications are used widely nowadays. They offer real-time video monitoring for homes, offices, warehouses, airports, and so on with live and pre-recorded on-demand video streaming. Quality of service (QoS) remains a key challenge faced by most of these applications. In this article, we propose an architecture for mobile video surveillance applications with a guaranteed and differentiated QoS support. The architecture relies on the 3GPP 4G evolved packet core (EPC). The main components are the QoS enabler, media server, and machine-to-machine gateway and surveillance application. To demonstrate its feasibility, a proof of concept prototype has been implemented and deployed. We also took measurements to evaluate the performance. Several lessons were learned. For instance, multimedia frameworks must allow for buffering controls in media streaming to reduce live streaming delay. In addition, we have learned that publicly available materials related to the EPC prototyping platform we have used (i.e., OpenEPC) are scarce. This has made our prototyping task rather difficult.

INDEX TERMS Video surveillance, 4G, evolved packet core (EPC), machine-to-machine (M2M).

I. INTRODUCTION

Mobile video surveillance applications are widely used for both civilian and military purposes. They provide remote monitoring and real-time information to improve public safety and asset protection. They are composed of source modules, functional modules and sink modules [1]. The source modules are generally machine-to-machine (M2M) devices (e.g., an IP camera and a motion detector) that sense events and capture video. The functional modules perform task processing on the captured video. They may transform the quality or extract specific information. The sink modules are the ultimate destination of the captured videos. They have input but do not have any output, except for sometimes, a human-machine interface or alarm actuators.

Video streaming in mobile video surveillance applications is sensitive to delay, jitter and packet loss. This sensitivity makes guaranteed and differentiated quality of service (QoS) a critical issue in mobile video surveillance applications. Guaranteed QoS ensures the bandwidth and upper bound on the delay and packet loss ratio for specific data flow. Differentiated QoS, in its simplest form, provides different levels of QoS to different traffic classes (e.g., voice vs. video).

Let us take the example of an avant-garde shopping center that offers a video surveillance application with different QoS classes to its merchants. Merchants can select their QoS class depending on how the merchants value what they have in their respective shops. They can also change the QoS class during the sessions. Several challenges can be derived from the example. The very first challenge is the ability to offer a refined QoS scheme to go far beyond the traditional coarse-grained QoS, which can differentiate only between different traffic classes (e.g., voice vs. video) by differentiating between the different end-users of the same application and even among different sessions of the same end-user. Yet another challenge is the need to cater to the heterogeneity of M2M devices. The third challenge is the necessity to use a non-customized underlying network in order to ease the application deployment. The novelty of the article, which extends our previous work presented at a conference [2], is that it proposes an architecture for mobile video surveillance applications that supports both guaranteed and session-based

differentiated QoS and that tackles these three challenges. To the best of our knowledge, no such architecture exists today.

We use 3GPP 4G evolved packet core (EPC) as the underlying network for our architecture. The contributions encompass an overall system architecture and a set of application enablers (i.e., QoS enabler, media server and M2M gateway) for the development and management of mobile video surveillance applications to offer refined, guaranteed and differentiated services in a heterogeneous M2M environment and a standard 4G EPC environment. The system architecture includes a set of procedures to provide preferential treatment for higher priority QoS classes at the application level.

TABLE 1. List of frequently used acronyms.

AF	Application Function
API	Application Program Interface
BBERF	Bearer Binding and Event Reporting Function
CoAP	Constrained Application Protocol
EPC	Evolved Packet Core
ePDG	Evolved Packet Data Gateway
IMS	IP Multimedia Subsystem
LTE	Long Term Evolution
LTE-A	Long Term Evolution-Advanced
M2M	Machine-to-Machine
PCC	Policy and Charging Control
PCEF	Policy and Charging Enforcement Function
PCRF	Policy and Charging Rule Function
PDN-GW	Packet Data Network Gateway
QoS	Quality of Service
REST	Representational State Transfer
RTP	Real-time Transport Protocol
RTSP	Real-time Streaming Protocol
S-GW	Serving Gateway
SOAP	Simple Object Access Protocol
SPR	Subscription Profile Repository
URI	Uniform Resource Identifier
VM	Virtual Machine

Table 1 lists the frequently used acronyms in this article. The rest of this article is organized as follows: Section II provides background information on EPC. Section III discusses related work. Section IV describes the proposed architecture. Section V presents our prototype and the performance measurements. The lessons learned are discussed in section VI. In section VII, we conclude our present research.

II. BACKGROUND INFORMATION ON EPC

EPC [3], [4] is a flat IP-based core network architecture for the long term evolution (LTE) wireless access technology. It can also accommodate other 3GPP access networks, such as GPRS and UTRAN, and even non-3GPP access networks, such as WiMAX and Wi-Fi. The new architecture enhances network performance by separating data and control paths. On the data path, the serving gateway (S-GW) and evolved packet data gateway (ePDG) act as the access gateways for LTE and the untrusted non-3GPP access networks respectively and the packet data network gateway (PDN-GW) acts as the gateway to the external networks and nodes. On the control path, the S-GW, ePDG and PDN-GW interact with the policy and charging rule function (PCRF) entity, which is considered the main component of the policy and charging control (PCC) system.

PCC enables the EPC operator to provide applications with differentiated QoS and different charging models [5]. The QoS in EPC is class-based, wherein each class provides a different packet forwarding treatment (e.g., scheduling policy). Each QoS class is associated with predefined characteristics in terms of the resource type (guaranteed or non-guaranteed bitrate resources), priority, packet delay budget (upper bound) and packet error loss rate (upper bound). The 3GPP Release 12 specifications include thirteen standardized classes with corresponding characteristics [4]. However, the EPC operators can still define new QoS classes, such as platinum, gold and silver used in this article to meet their business need.

The EPC architecture shown in Figure 1 depicts a simplified version of the 3GPP Release 12 PCC architecture without the charging functional components. The main functional entities are the application function (AF), PCRF, the subscription profile repository (SPR), the policy and charging enforcement function (PCEF) and, if applicable, the bearer binding and event reporting function (BBERF). BBERF entity is used in PCC architecture when a mobility protocol other than GPRS tunneling protocol (GTP) is used in EPC [6].

AF is an application server that provides services (e.g., video surveillance) to end-users. It sends the session requirements (e.g., QoS class) to PCRF via the Rx reference point to reserve network resources. PCRF, in turn, creates the PCC rules and pushes them towards PCEF and BBERF (if applicable) via the Gx and Gxx (i.e., Gxb and Gxc) reference points, respectively. PCRF may retrieve the user subscription information from SPR to create the PCC rules. PCEF and BBERF are the enforcement entities that reside in PDG-GW and the access gateway (e.g., S-GW and ePDG), respectively. They enforce the PCC rules and notify PCRF for the events related to data traffic, such as the radio access technology change. The interested reader may refer to [7] for a detailed tutorial on the 3GPP 4G EPC.

III. RELATED WORK

Next, we review works related to the system architecture, the QoS enabler, and M2M communication. On the system architecture front, Carpenter and Nichols [8] propose a high-level architecture to offer QoS over the Internet. However, it does not provide a concrete solution to allow the applications to negotiate QoS and support session-based

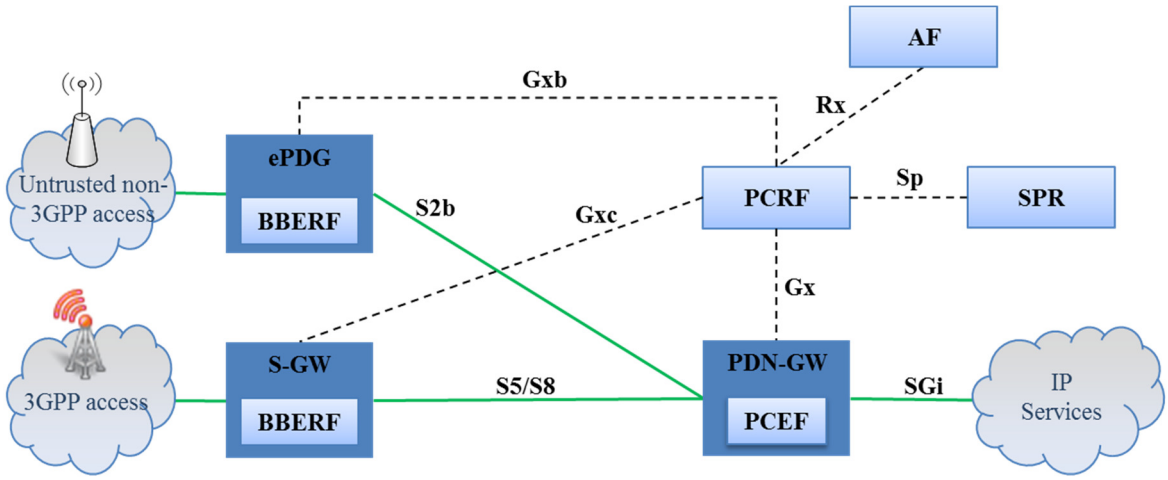

FIGURE 1. Simplified 3GPP EPC architecture.

differentiated QoS. García et al. [9] propose a QoS control mechanism for providing service differentiation over the Internet. The proposed architecture handles QoS at the server side, but does not tackle the challenge of providing QoS over the Internet. RFC 2990 [10] outlines the most outstanding IP QoS architectural issues on the Internet that have impeded a wide scale commercial deployment of the Internet QoS. However, it does not propose a concrete solution.

The studies [11] and [12] propose architectures for mobile video surveillance applications that use the Internet and mobile networks to interconnect system components and provide remote access. However, they do not tackle the QoS challenge. In fact, the proposed architectures use the adaptive bitrate streaming to adapt the streaming to the network bandwidth variation to provide the best possible quality for the video streaming. Lin et al. [13] propose an architecture for mobile video surveillance that relies on an IP multimedia subsystem (IMS) to support QoS. Thus, the architecture supports guaranteed QoS but not session-based differentiated QoS. El Barachi et al. [14] propose an architecture for call differentiation for 3G networks. The architecture extends the IMS architecture to enable guaranteed and session-based differentiated QoS. The main drawback is that it requires the addition of new functional entities to IMS as well as the enhancement of existing ones.

On the QoS enabler front, some application programming interfaces (APIs) have been proposed in the literature. However, these APIs do not generally incorporate support for session-based differentiated services. Implementation is also not usually covered. An example is [15], which proposes simple object access protocol (SOAP) web service APIs to support QoS. However, the APIs do not support session-based differentiated services. The same applies to the QoS APIs proposed by [16], which uses IMS to support the QoS provisions for third-party service providers.

The work in [17] is dedicated to M2M communication over 3GPP LTE and long term evolution-advanced (LTE-A) systems. In [18], the authors survey the platforms for M2M applications. Both papers detail the challenges related to M2M communication and present possible enabling technologies. They discuss the QoS at the M2M nodes level (e.g., battery efficiency), access (e.g., radio resource sharing), or network layer (e.g., jitter). However, none of the two papers discuss an architecture for QoS-enabled applications that supports both guaranteed and session-based differentiated QoS, which is the focus of this paper.

IV. PROPOSED ARCHITECTURE

We first discuss the system architecture. Then we present the interfaces and functional components of the architecture, followed by the procedures. An illustrative scenario is presented after that.

A. SYSTEM's ARCHITECTURE

Figure 2 depicts the system architecture, comprising three domains: M2M, network and application domains. The M2M domain encompasses heterogeneous M2M devices that offer the surveillance services such as video capturing and motion detection. These devices are connected to the M2M gateway, which hides the complexity and heterogeneity of the M2M domain from the rest of the system. The M2M devices are connected to the gateway through the M2M network. Although not shown in the figure, there might be several M2M networks to cater to the heterogeneity of M2M communications and networking technologies (e.g., IP-based vs. non-IP-based). The M2M gateway supports various M2M communication and networking technologies (e.g., ZigBee and 6LowPAN), application protocols (e.g., constrained application protocol [CoAP] and SOAP), and addressing schemes (e.g., ZigBee and IP) at the southbound interface to interact with the M2M devices. At the northbound interface, it exposes the M2M devices capabilities via unified API to the M2M applications (i.e., surveillance applications in our case). In our view, this gateway

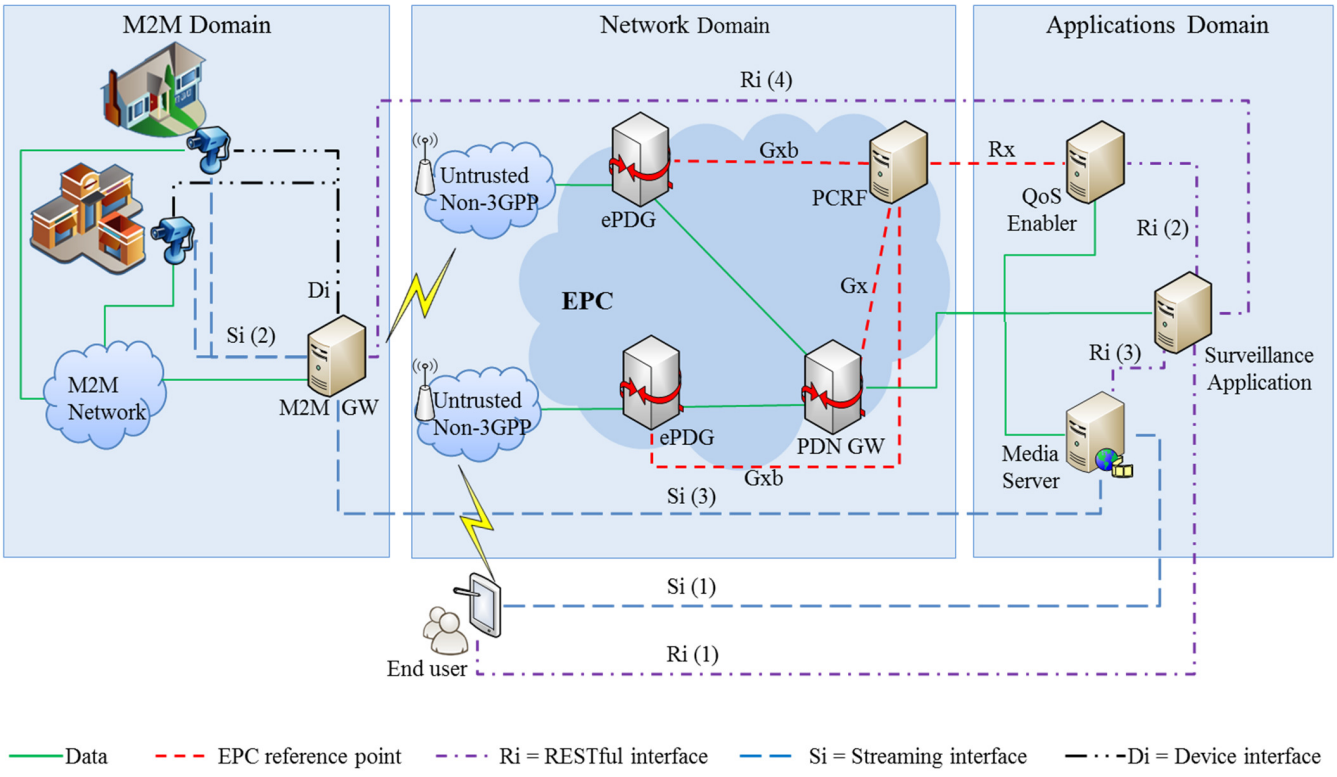

FIGURE 2. System's architecture.

driven approach we are stipulating in this article is much more realistic than accessing directly M2M devices from the EPC domain. This is due to the fact that the landscape of M2M communications, networking and application standards is highly heterogeneous and constantly evolving.

EPC represents the network domain in our architecture. It is used as the underlying network because it offers more support for QoS compared to its alternatives, such as the Internet and the 3GPP 3G networks. In our architecture, EPC provides the connectivity between the M2M gateway and the surveillance application as well as between the surveillance application and the end-users. It ensures the guaranteed and session-based differentiated QoS at the network layer. More precisely, as per the business agreement, EPC offers the surveillance application a set of guaranteed QoS classes (e.g., platinum, gold and silver) so that surveillance application can request assigning any of these offered classes to its video streaming sessions. The surveillance application exposes these QoS classes to its end-users allowing them to choose the desired QoS class for each session. This enables the guaranteed and session-based differentiated QoS that differentiates between the end-users' sessions of the same application and even among different sessions of the same end-user. This refined QoS scheme does not require any changes in EPC.

In the application domain, the QoS enabler exposes the EPC network QoS capabilities to the surveillance application and any third-party service providers. The media server enables live and on-demand video streaming with different video qualities (e.g., high, medium and low quality) as well as video recording for later viewing. The surveillance application offers the end-users the surveillance service that supports video streaming with guaranteed and session-based differentiated QoS. The end-users access this service via a surveillance client application installed on their user equipment devices. To offer such service, the surveillance application uses the northbound interface of the M2M gateway to access the surveillance capabilities offered by the M2M domain This interface is accessed via the addressing scheme (i.e., IP-based) offered by 3GPP EPC. Furthermore, the surveillance application leverages the QoS classes of EPC and the QoS enabler API to support the guaranteed and session-based differentiated QoS at the network layer. It also implements a session admission procedure to provide preferential treatment for higher priority QoS classes at the application level, as will be discussed in section IV.D.

B. INTERFACES

The main interfaces used in our architecture as shown in Figure 2 are:

- Ri (RESTful interfaces): We use RESTful web services as technology for the interfaces of our architecture's main components (i.e., QoS enabler, media server, M2M gateway and surveillance application) since it has

TABLE 2. M2M gateway RESTful resources.

Resources	Base URL: http://{serverRoot}/{apiVersion}/m2m/{applicationId}	HTTP action
The list of M2M devices that can be accessed by a specific surveillance application	/devices	GET: return all M2M devices POST: add a new M2M device
A specific M2M device	/devices/{deviceId}	GET: return specific M2M device information PUT: update specific M2M device information DELETE: remove a specific M2M device
The list of fired M2M events (e.g., motion detection) for a specific surveillance application	/events/	GET: return all M2M events
A specific fired M2M event	/events/{eventId}	GET: return specific M2M event information DELETE: remove a specific M2M event
The list of video streaming sessions for a specific application	/streams	GET: return all ongoing video streaming sessions POST: establish a new video streaming session
A specific video streaming session	/streams/{streamId}	GET: return a specific video streaming session information PUT: update a specific video streaming session (e.g., video quality) DELETE: terminate a specific video streaming session
The list of M2M event notification subscription profiles for a specific application	/subscription	GET: get all subscription profiles for a specific application (Note: each application can have at most one subscription profile) POST: create a new subscription profile
A specific M2M events notification subscription profile	/subscription/{subscriptionId}	PUT: update a specific subscription profile DELETE : remove a specific subscription profile
Callback resource (client side resource)	URI provided by the client at subscription time	POST: post the M2M event notification information to the client

many benefits compared to SOAP web services when used in resource-constrained environments (i.e., the mobile phones/devices used by end-users in our case). These benefits include less overhead, less parsing complexity and statelessness [18]. RESTful web services model the information as resources that are identified using uniform resource identifiers (URIs) and are accessible via HTTP methods POST, GET, PUT and DELETE to create, read, update and delete a resource, respectively. Reference [19] gives more details on RESTful web services.

We define four RESTful interfaces: Ri (1), Ri (2), Ri (3) and Ri (4). All interfaces define server-side RESTful resources to expose the capabilities of the servers implementing them. Ri (1), Ri (2) and Ri (4) interfaces also define a client-side resource, known as the callback resource. The client provides the server with the URI of the callback resource in order to receive a notification when an event occurs. The Ri (1) interface is used by the end-users to access the surveillance service. The surveillance application uses the Ri (2) interface to access the QoS enabler capabilities to reserve, update and release the network resources for video streaming sessions. It also uses the Ri (3) interface to access the media server functionalities, such as video recording. The M2M gateway exposes its capabilities toward the surveillance application via the Ri (4) interface. These capabilities include, but are not limited to, creating a video streaming session and subscribing to an event notification (e.g., motion detection alarm). As an example of these interfaces, the QoS enabler interface was presented in our conference paper [2], extended in this article. Furthermore, TABLE 2 summarizes the proposed RESTful resources for the M2M gateway.

- Di (M2M device interface): This interface is used between the M2M devices (e.g., IP cameras) and the M2M gateway. It is used to manage and access the M2M devices. It is dependent on the used devices and may differ from one device to another. For instance, the M2M devices used in our implementation support the SOAP interface.
- Si (Streaming interfaces): The Si interfaces are used for video streaming over the network. The Si (1) interface is used between end-users and the media server. It uses real-time streaming protocol (RTSP)/real-time transport protocol (RTP). RTSP is a client-server protocol used to establish and control media sessions. RTP is a transport protocol for real-time media delivery over networks. The Si (2) interface is used for video streaming from the M2M devices to the M2M gateway. It is dependent on the M2M devices. For instance, the M2M devices used in our implementation support RTSP/RTP. The Si (3) interface uses RTP to stream the video between the M2M gateway and the media server.
- Rx and other interfaces in the network domain: They are 3GPP standard reference points. Rx runs between

an AF and PCRF in the EPC architecture. We used it to exchange application level session information between the QoS enabler and PCRF in order to reserve, update and release network resources. The other interfaces are used as per the standard specifications.

C. FUNCTIONAL COMPONENTS' ARCHITECTURES

This section gives more details about the main components of our architecture (i.e., the QoS enabler, media server, surveillance application and the M2M gateway). All these components share three architectural layers, namely: basic capability, service and RESTful API layers. The basic capability layer provides the essential functions and capabilities (e.g., communication and media processing) required to build the service layer. The service layer provides the core services and functionalities of the component. The API layer exposes these services and functionalities via RESTful API.

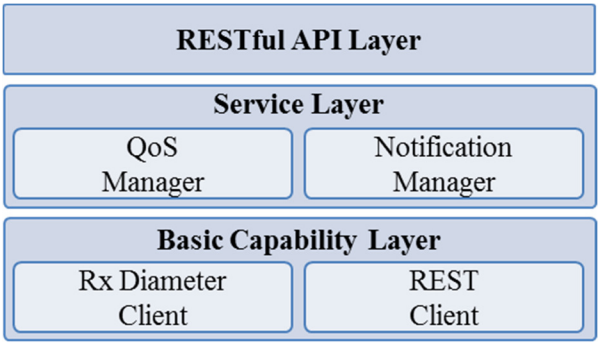

FIGURE 3. QoS enabler architecture.

Figure 3 shows the proposed architecture of the QoS enabler. It relies on the diameter client, which implements the Rx diameter interface, to communicate with PCRF. The REST client is used to send notifications and session termination requests to the surveillance application. The QoS manager receives the surveillance application's requests to create, modify and delete sessions; it validates the request message content and checks if the surveillance application is authorized to request the desired QoS class. Furthermore, it communicates with PCRF via the Rx diameter client in order to send session information and receive QoS event notifications. The notification manager sends event notifications (e.g., resource reservations, bearer release events and QoS changes) to the surveillance application.

Figure 4 shows the surveillance application architecture. It relies on a REST client to communicate with the QoS enabler, the media server, the M2M gateway and the end-users. The notification manager manages the notification subscriptions of end-users and notifies them when an alarm is fired. The session manager processes the streaming requests and communicates with the QoS enabler, the media server and the M2M gateway to create, modify and teardown the streaming sessions. The M2M manager maintains a list of registered M2M gateways and M2M devices.

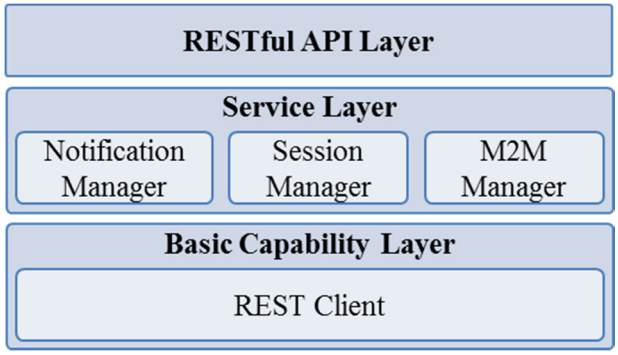

FIGURE 4. Surveillance application architecture.

Furthermore, it communicates with the M2M gateways to determine the M2M devices' statuses and capabilities.

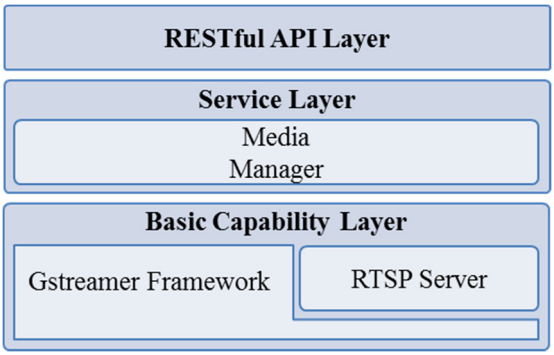

FIGURE 5. Media server architecture.

Figure 5 depicts the media server architecture. The GStreamer framework[1] is an open source framework for developing multimedia applications. It allows real-time media capturing, processing, transcoding and transrating. The Gstreamer RTSP server[2] is a streaming server based on the Gstreamer framework. It delivers live and on-demand video using the standard protocols RTSP/RTP. The media manager (at the service layer) creates media sources to capture media from RTP flows or from files and then it processes the captured media to generate video according to the desired video quality (i.e., low, medium and high). It also creates media sinks to deliver media to the end-user via the RTSP server or to record a video for deferred viewing.

Figure 6 depicts the proposed architecture of the M2M gateway including the most relevant interactions between the component's modules as well as the interactions with other components in the system architecture. The M2M gateway encompasses two layers dedicated to handling the interactions with the M2M devices, namely: protocol stack and device abstraction layers. The protocol stack layer encompasses the protocols stacks (e.g., ZigBee and CoAP) required to trigger, access and manage the heterogeneous

[1] http://gstreamer.freedesktop.org/
[2] http://gstreamer.freedesktop.org/modules/gst-rtsp-server.html

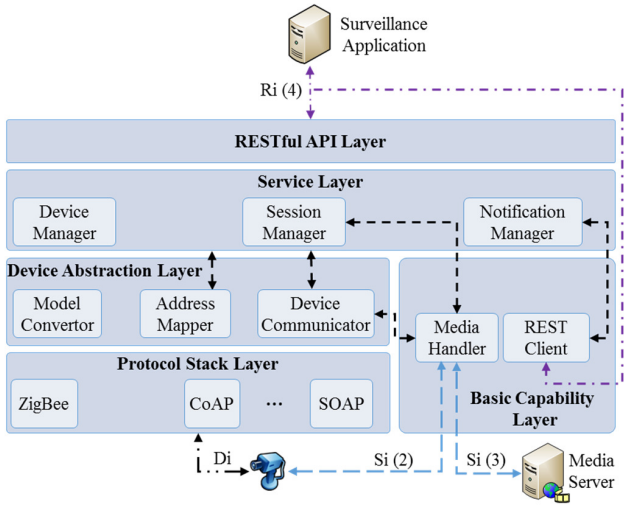

FIGURE 6. M2M gateway architecture.

M2M devices. The device abstraction layer hides the heterogeneity of M2M devices technology from the service layer. It encompasses three main modules: model convertor, address mapper and device communicator. The model convertor transforms the data models (independent of M2M technology) used in the service layer to M2M technology models and vice versa. The address mapper performs the mapping between the application-level address of the M2M device (e.g., unique Id) and the address used in the underlying M2M technology. The device communicator is used to interact with the M2M devices independently of the underlying M2M technology. It provides the capabilities required to communicate with the M2M devices such as sending and receiving messages.

At the basic capability layer, the media handler uses the Gstreamer framework to capture the video streaming from the M2M device and to transmit it to the media server. The REST client is used to send alarm notifications to the surveillance application. In addition, at the service layer, the device manager is responsible for the management of the M2M devices (e.g. configuration and monitoring). It also maintains the list of M2M devices, their statuses and capabilities. The session manager is responsible for managing the streaming sessions between the M2M devices and the M2M gateway as well as between the M2M gateway and the media server. The notification manager manages the notification subscriptions of the surveillance applications and subscribes to the alarm notification service of the M2M devices. Moreover, it notifies the surveillance applications when an alarm is received from the M2M devices.

D. PROCEDURES

Using the proposed architecture, end-users can initiate sessions with a specific QoS class (e.g., platinum, gold and silver) and a video profile (e.g., high, medium and low quality) based on their preferences. They can also upgrade or downgrade the QoS class or video profile of their ongoing sessions. We first present the main parameters used for session management, and then we discuss the various related procedures.

1) PARAMETERS

We assume that each surveillance service provider is assigned a limited bandwidth by the EPC provider (as per their agreement). For each QoS class, we define the guaranteed bandwidth parameter configurable by the surveillance service provider, such that the summation of the guaranteed bandwidth for all QoS classes equals to the bandwidth assigned to the surveillance service provider. High-priority QoS classes have higher guaranteed bandwidth. This parameter provides higher-priority QoS classes with a better admission rate. For the application as a whole, we define the saturation threshold (e.g., 90%). If the total bandwidth used by the application is beyond the configured saturation threshold, a new session is accepted only if the total bandwidth currently used by the target QoS class sessions is below the guaranteed bandwidth of the QoS class.

2) SESSION ADMISSION PROCEDURE

When the application receives a request for a new session with a given QoS class, it checks if: (1) the QoS class is currently using its entire guaranteed bandwidth; and (2) the application saturation threshold is reached. If this is the case, the new session is rejected. Otherwise, the application checks if enough resources are available. If yes, the session is established. If not, the application tries to free up the necessary bandwidth (using the procedure below) and then admits the new session if enough resources are released.

3) BANDWIDTH RELEASE PROCEDURE

The application uses video profile downgrading and session termination techniques to free up bandwidth. Session termination is used only if: (1) it is not possible to free the required bandwidth using session downgrading; and (2) the bandwidth used by the target QoS class after admitting this session doesn't exceed the QoS class guaranteed bandwidth. The application releases the bandwidth such that the total bandwidth currently used by each QoS class does not become less than the guaranteed bandwidth for each QoS class.

4) GUARANTEED AND SESSION-BASED DIFFERENTIATED QoS

Figure 7 depicts the main steps of provisioning guaranteed and session-based differentiated QoS during the session establishment. The figure shows, as an example, that EPC authorizes the surveillance application to use the QoS classes: platinum, gold and silver. We assume that these classes offer guaranteed QoS. To illustrate the guaranteed and session-based differentiated QoS, the figure also shows three ongoing video streaming sessions with different QoS classes for the same end-user. Moreover, the end-user uses the surveillance client application installed on the user equipment device to

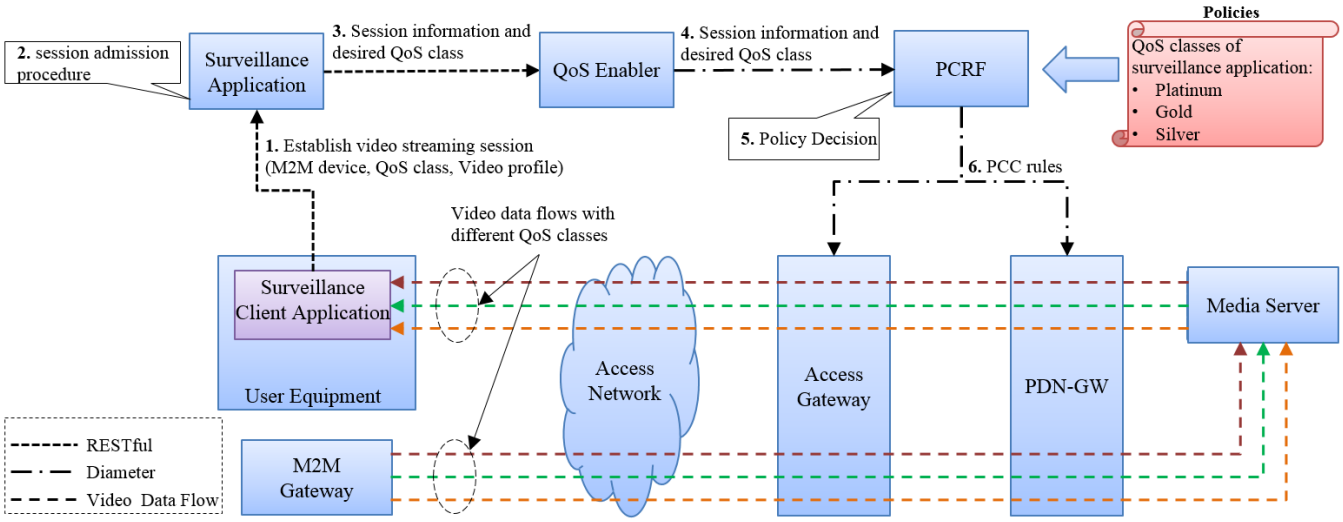

FIGURE 7. Guaranteed and session-based differentaited QoS provisioning.

access the surveillance service. When an end-user places a request to establish a video streaming session, the request will include the source M2M device, the desired QoS class and desired video profile (step 1). The QoS class can be any of the QoS classes supported by the surveillance application. On receiving the request, the surveillance application runs the session admission procedure to admit or reject the session establishment (step 2). If the request is accepted, it sends a request to the QoS enabler to authorize the session (step 3). The QoS enabler uses Rx reference point to communicate with PCRF to authorize the session and reserve the required network resources. If the request is compliant with the policies, PCRF admits the session (step 5). Then , it translates the session information, which includes the desired QoS class, into the appropriate QoS parameters (e.g., guaranteed bitrate) and builds the PCC rules. After that, it pushes the PCC rules to the PCEF and BBERF (if applicable) enforcement entities residing in the PDN-GW and access gateway respectively (step 6). The enforcement entities ensure that the packets receive the appropriate QoS treatment at the network layer.

5) OTHER PROCEDURES

The procedure to upgrade an ongoing session is similar to that of session admission. To downgrade a session, the application communicates with the PCRF via the QoS enabler to downgrade the reserved network resources, as defined in the EPC specification.

E. ILLUSTRATIVE SCENARIO

We assume that an avant-garde shopping center offers a video surveillance application to its merchants. The application supports three QoS classes, namely: platinum, gold and silver (ordered from a higher QoS class to a lower one). It also supports the video streaming with three video quality profiles: high, medium and low. Alice and Bob are two merchants who have subscribed to the surveillance application and who own a candy store and a jewelry store, respectively. Alice's and Bob's stores are located in proximity to each other.

Figure 8 shows how the illustrative scenario can be realized using the proposed architecture. In this scenario, Bob starts live video streaming from the camera near his store using silver QoS class and medium quality video profile (steps 1 to 14). During Bob's session, an intrusion is detected in a common area close to both Alice's and Bob's stores. The surveillance application starts video recording from the camera and informs the end-users who have subscribed to the intrusion event, including Alice and Bob (steps 15 to 30). When Alice receives the notification, she decides to start a high-quality video streaming using the gold QoS class (steps 31 to 38). Soon after, Bob decides to upgrade his ongoing streaming session to the platinum QoS class in order to achieve smoother streaming service and he also upgrades the session to high-quality video to attain a better streaming image (steps 39 to 46).

V. IMPLEMENTATION

A. PROTOTYPE

As a prototype, we partially implemented the scenario given in section IV.E. We implemented the alarm notification, session creation and QoS class upgrading /downgrading, but we did not implement the video quality upgrading/downgrading functionalities. The REST interfaces of the different components were implemented using the Restlet framework. The Rx diameter client interface of the QoS enabler was implemented using the JavaDiameterPeer library.

We used Fraunhofer Fokus OpenEPC[3] Release 2 as the 3GPP 4G EPC. In the M2M domain, we used three AXIS M1031-W network cameras as M2M devices,

[3]http://www.openepc.net/index.html

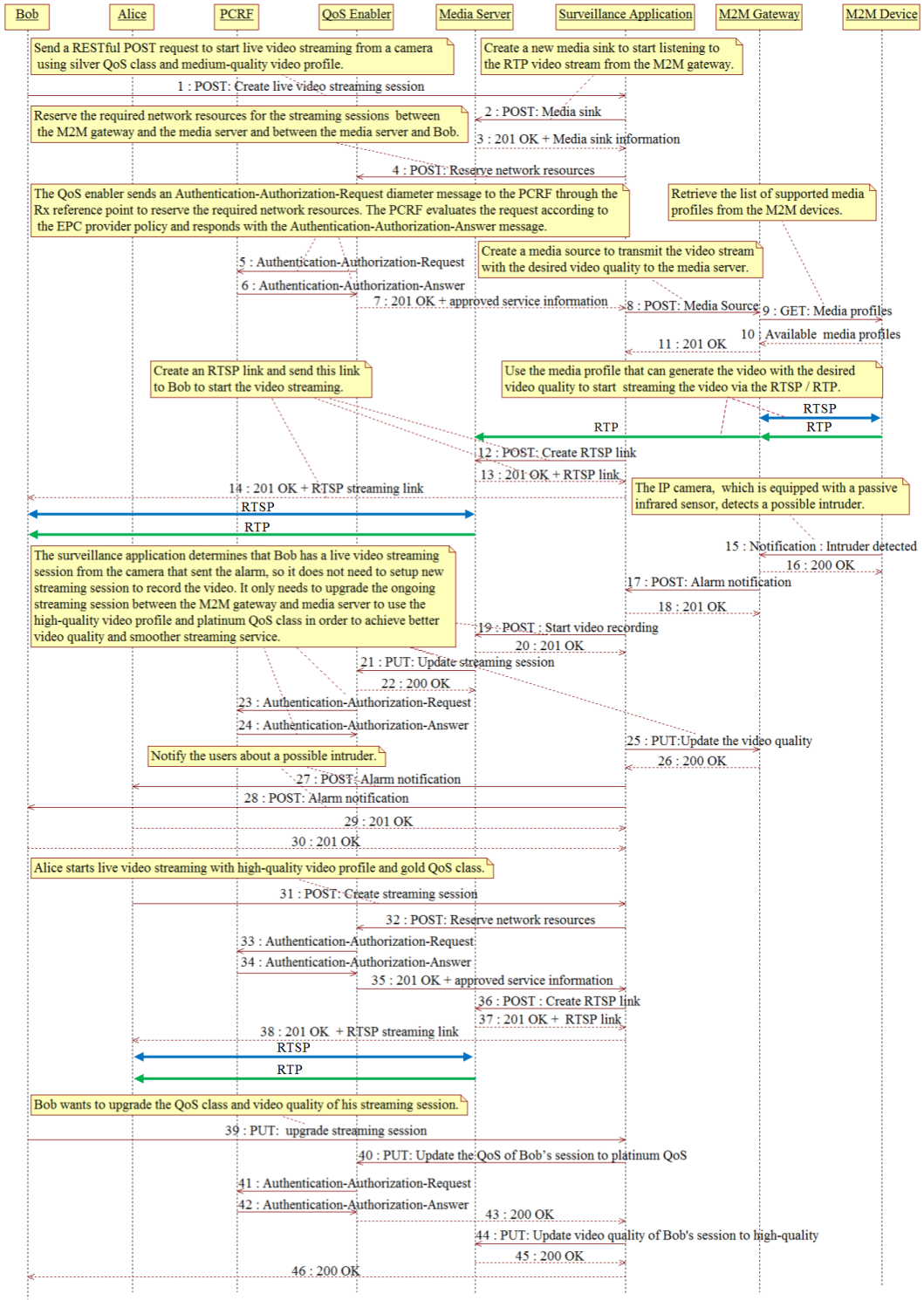

FIGURE 8. Illustrative scenario.

each equipped with a passive infrared sensor. The cameras communicate with the M2M gateway using a SOAP web service, implemented using JAX-WS. A simple version of the M2M gateway was implemented. This version exposed the video streaming and motion detection notification capabilities of the M2M devices to the surveillance application.

To stay informed about new events captured by the camera, the M2M gateway subscribes to the camera using the Apache muse implementation of the WS-BaseNotification. WS-BaseNotification is a SOAP web service standard created to publish and subscribe to event notifications. The M2M gateway also uses the Gstreamer framework to capture the video stream from the M2M devices and transmit it to the media server.

The surveillance client application is an Android application developed using Android 3.0 SDK and deployed on an ASUS Eee Pad Transformer tablet. It uses a Restlet client API to communicate with the surveillance application, and it also hosts an internal Restlet HTTP server connector to receive the alarm notifications from the surveillance application. The internal Restlet HTTP server connector is a compact, lightweight HTTP server with no external dependency.

The prototype setup consists of five virtual machines (VMs) that host the M2M gateway, OpenEPC, QoS enabler, media server and the surveillance application. The VMs run on an Intel core i7 laptop with 8 GB of RAM. The prototype used two ePDG nodes as the access gateways for EPC. One is connected to the M2M gateway via a virtual network. The M2M gateway VM is connected to a Wi-Fi network that represents the M2M network. The other ePDG is connected to another Wi-Fi that represents an access network used by the end-user's tablet to connect to EPC.

B. PERFORMANCE ANALYSIS

The prototype's performance is evaluated in terms of session creation delay, end-to-end session creation delay and session update delay. The session creation delay is measured as the difference between the time when the surveillance client application sends the request to start a live video streaming session and the time at which it receives the response that contains the RTSP link needed to start the video streaming. This delay does not include the client-side media setup. The end-to-end session creation delay is measured as the difference between the time when the surveillance client application sends the request to start the live video streaming and the time at which the video stream is played on the end-user's tablet. In other words, it consists of the session creation delay, the time needed for the RTSP communication between the surveillance client application, the media server and the Android media player buffering time. Finally, the session update delay is measured as the difference between the time when the surveillance client application sends the request to the surveillance application to update the QoS profile for the current ongoing session and the time at which it receives the confirmation response.

Figure 9 shows the evaluation results. The session creation average delay is 1036.9ms, which is barely noticeable by the end-users. The end-to-end session creation average delay is 1483.3ms. This means that the client-side media setup takes 446.4ms. The media setup delay includes the RTSP communication and the Android media player buffering time. The media setup delay can be improved by using a media player that provides API to control the media buffering time. The current prototype uses the default Android 3.0 media player, which does not provide this feature. The figures also show that the average time needed to update the QoS profile for an ongoing session is 618.7ms, which is not noticeable by end-users.

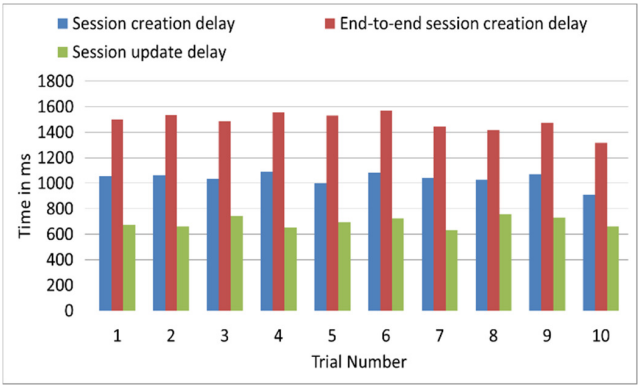

FIGURE 9. Performance measurements.

VI. LESSONS LEARNED

The first lesson we learned concerns the media processing in the Gstreamer framework. It has pipeline-based architecture, in which the media pipeline consists of a sequence of elements such that the output of one element is an input of the next element. The pipeline must start with a source element and end with a sink element. The source element generates media data for use by the pipeline. The sink element is the terminal point in the pipeline. It receives the media data from the pipeline and sends it to its final destination, such as an audio card, an RTP socket or a file (i.e., media data saved into a file). Each element between the source and sink elements implements a specific media processing function (e.g., encoding and mixing). The direct connection between elements enables media data to continuously transfer through the pipeline's elements rather than waiting for one element to finish before the next one can start, which is important in live video streaming because it allows for on-the-fly media processing. Besides, Gstreamer has a tool called ''gst-launch'' that allows the developers to build, run and test basic Gstreamer pipelines, simplifying the development of streaming media applications.

The second lesson is about media buffering in live video streaming. Buffers are an essential part of creating, processing, delivering and rendering media content. The larger the media buffer size, the longer delay there will be in the live video stream, which could be unacceptable, especially for critical applications such as mobile video surveillance. Therefore, it is important that media frameworks give the developers the ability to control the media buffering according to their requirements in order to minimize the streaming delay. The Gstreamer framework used to implement the M2M gateway and media server offers such possibility, which is not

the case for the Android 3.0 SDK used on the mobile handset side. Therefore, we believe that it would be worthwhile to extend the Android SDK to provide an API for developers to control buffering time, especially for live streaming. An alternative could be to use another API that supports media buffering controlling on the client's side. An example would be Gstreamer, which has recently added support for the Android operating system.

The third lesson is about the API of the surveillance IP cameras. The M2M gateway exposes the surveillance IP cameras' capabilities to surveillance applications. It supports surveillance IP cameras from different vendors with different capabilities. A standardized interface will enable the interoperability of IP-enabled surveillance cameras and make building M2M gateways easier. However, the security industry, the manufacturer of surveillance IP cameras, lacks such an interface. The security industry has two specifications for IP surveillance and security devices, including surveillance IP cameras: Physical Security Interoperability Alliance (PSIA) and Open Network Video Interface Forum (ONVIF). PSIA adopts REST architecture, whereas ONVIF uses SOAP web service standards.

Finally, the availability of EPC implementation is essential for the development and testing of EPC-related prototypes and applications. However, the available choices are limited to OpenEPC, which is developed by Fraunhofer FOKUS. The OpenEPC is a non-open source software implementation of 3GPP EPC, which enables the prototyping of IP connectivity features like QoS and mobility management. Due to the complexity of EPC and its implementation, OpenEPC has a steep learning curve, especially when it comes to reconfiguring the system (e.g., adding new ePDG to the system), which is not explained in the high-level documentation shipped with it. OpenEPC usage is limited to researchers in universities and telecommunication companies, making the materials related to OpenEPC on the Internet very limited. Therefore, it would be worth having a discussion group and a mailing list for OpenEPC to speed-up the learning curve.

VII. CONCLUSION

We propose a novel architecture for the mobile video surveillance applications over EPC. The architecture provides for the rapid development and deployment of new applications, combined with guaranteed and differentiated QoS, which is not possible with other networks. A proof of concept prototype was successfully implemented; as demonstrated by the performance measurement analysis, the delays incurred by using the proposed architecture are acceptable from the end-user's point of view.

ACKNOWLEDGMENT

This paper was presented at the 2012 third international conference on the Network of the Future under the title A 3GPP 4G Evolved Packet Core-Based System Architecture for QoS-Enabled Mobile Video Surveillance Applications.

REFERENCES

[1] R. Cucchiara and G. Gualdi, "Mobile video surveillance systems: An architectural overview," in *Mobile Multimedia Processing*, X. Jiang, M. Y. Ma, and C. W. Chen, Eds. Heidelberg, Germany: Springer, 2010, pp. 89–109.
[2] M. Abu-Lebdeh, F. Belqasmi, and R. Glitho, "A 3GPP 4G evolved packet core-based system architecture for QoS-enabled mobile video surveillance applications," in *Proc. 3rd Int. Conf. Netw. Future (NOF)*, Tunis, Tunisia, 2012, pp. 1–6.
[3] *General Packet Radio Service (GPRS) Enhancements for Evolved Universal Terrestrial Radio Access Network (E-UTRAN) Access, Version 12.9.0, Release 12*, document TS 23.401, 3GPP, Jul. 2015.
[4] *Architecture Enhancements for Non-3GPP Accesses, Version 12.8.0, Release 12*, document TS 23.402, 3GPP, Apr. 2015.
[5] *Policy and Charging Control Architecture, Version 12.9.0, Release 12*, document TS 23.203, 3GPP, Jul. 2015.
[6] J.-J. P. Balbas, S. Rommer, and J. Stenfelt, "Policy and charging control in the evolved packet system," *IEEE Commun. Mag.*, vol. 47, no. 2, pp. 68–74, Feb. 2009.
[7] M. Olsson and C. Mulligan, *EPC and 4G Packet Networks: Driving the Mobile Broadband Revolution*, 2nd ed. San Diego, CA, USA: Academic, 2013.
[8] B. E. Carpenter and K. Nichols, "Differentiated services in the Internet," *Proc. IEEE*, vol. 90, no. 9, pp. 1479–1494, Sep. 2002.
[9] D. F. García *et al.*, "A QoS control mechanism to provide service differentiation and overload protection to Internet scalable servers," *IEEE Trans. Services Comput.*, vol. 2, no. 1, pp. 3–16, Jan./Mar. 2009.
[10] G. Huston, *Next Steps for the IP QoS Architecture*, document RFC 2990, IETF, 2000.
[11] X. Zhang and R. Lin, "Design and implementation of a mobile video surveillance system," in *Proc. Int. Conf. Internet Technol. Appl. (iTAP)*, 2011, pp. 1–5.
[12] W. Jian, C. Qimei, Z. De, and B. Houjie, "Embedded wireless video surveillance system for vehicle," in *Proc. 6th Int. Conf. ITS Telecommun.*, 2006, pp. 1141–1144.
[13] J. Lin, W. Lei, and Y. Liu, "The design of IMS based video surveillance system," in *Proc. 9th Int. Conf. Hybrid Intell. Syst. (HIS)*, vol. 2. 2009, pp. 283–288.
[14] M. El Barachi, R. Glitho, and R. Dssouli, "Control-level call differentiation in IMS-based 3G core networks," *IEEE Netw.*, vol. 25, no. 1, pp. 20–28, Jan./Feb. 2011.
[15] J. Y. Kim, J. H. Hahm, Y. S. Kim, and J. K. Choi, "Policy-based QoS control architecture model using API for streaming services," in *Proc. Int. Conf. Netw., Int. Conf. Syst., Int. Conf. Mobile Commun. Learn. Technol. (ICN/ICONS/MCL)*, 2006, p. 102.
[16] E. Pencheva and I. Atanasov, "Third party application control on quality of service in IP based multimedia networks," *Inf. Syst. Frontiers*, vol. 14, no. 3, pp. 555–569, 2012.
[17] F. Ghavimi and H.-H. Chen, "M2M communications in 3GPP LTE/LTE-A networks: Architectures, service requirements, challenges, and applications," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 525–549, 2nd Quart., 2015.
[18] J. Kim, J. Lee, J. Kim, and J. Yun, "M2M service platforms: Survey, issues, and enabling technologies," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 61–76, 1st Quart., 2014.
[19] Z. Shelby, "Embedded Web services," *IEEE Wireless Commun.*, vol. 17, no. 6, pp. 52–57, Dec. 2010.
[20] F. Belqasmi, R. Glitho, and C. Fu, "RESTful Web services for service provisioning in next-generation networks: A survey," *IEEE Commun. Mag.*, vol. 49, no. 12, pp. 66–73, Dec. 2011.

MOHAMMAD ABU-LEBDEH received the B.Sc. degree in computer engineering from An-Najah National University, Palestine, and the M.Sc. degree in electrical and computer engineering from Concordia University, Canada, where he is currently pursuing the Ph.D. degree in information and systems engineering. Prior to joining the M.Sc. program, he worked as a Software Engineer for several years. His current research interests include cloud computing, network functions virtualization, service engineering, and next generation networks.

FATNA BELQASMI received the M.Sc. and Ph.D. degrees in electrical and computer engineering from Concordia University, Canada. She was a Research Associate with Concordia University and a Researcher with Ericsson Canada. She was part of the IST Ambient Network project (a research project sponsored by the European Commission within the Sixth Framework Programme—FP6). She worked as an R&D Engineer with Maroc Telecom, Morocco. She is currently an Assistant Professor with Zayed University, Abu Dhabi, United Arab Emirates. Her research interests include next generation networks, service engineering, distributed systems, and networking technologies for emerging economies.

ROCH GLITHO received the M.Sc. degree in business economics from the University of Grenoble, France, in pure mathematics from the University of Geneva, Switzerland, and in computer science from the University of Geneva, and the Ph.D. (Tekn.Dr.) degree in teleinformatics from the Royal Institute of Technology, Stockholm, Sweden. He has worked in industry and has held several senior technical positions (e.g., Senior Specialist, Principal Engineer, and Expert) with Ericsson in Sweden and Canada. He is currently an Associate Professor and the Canada Research Chair with Concordia University, Montreal, Canada. He is also an Adjunct Professor with several other universities, including Telecom Sud Paris, France and the University of Western Cape, South Africa. His industrial experience includes research, international standards setting, product management, project management, systems engineering, and software/firmware design. He has also served as the IEEE Distinguished Lecturer, the Editor-in-Chief of the *IEEE Communications Magazine*, and the Editor-in-Chief of the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS journal.