A Framework for Interfaces of Trust and Security in the Mobile Edge Computing

Evelina Pencheva[1], Ivaylo Atanasov[1], Pencho Kolev[2], Miroslav Slavov[2]
[1]Technical University of Sofia
8 Kl. Ohridski Blvd, Sofia 1000
Bulgaria
{enp, iia}@tu-sofia.bg}

[2]Technical University of Gabrovo 4 Hadji Dimitar
Gabrovo 5300, Bulgaria
{pkpen@tugab.bg; miroslav)slavov@gmail.com}

ABSTRACT: *In order to enhance the network flexibility and decrease the time to market new services, mobile edge computing has been developed. In this area, cloud capabilities are increased in the radio networks. We have developed a framework of web service for trust and security management. The new web service permit MEC application registration and finding new services and initiating service agreement and also to control the internal load management.*

Keywords: Mobile Edge Computing, Infrastructure Services, Web Services, Service Oriented Architecture, Authentication

DOI: 10.6025/jisr/2020/11/4/105-112

Received: 1 May 2020, Revised 9 July 2020, Accepted 8 August 2020

1. Introduction

Mobile communications evolve continuously in order to meet the requirements for low latency, increase of traffic volumes, higher data rates and reliable connectivity. Mobile Edge Computing (MEC) is an emerging technology aimed to meet these requirements [1]. It exploits a combination of virtualization, cloud capabilities and coordination techniques which interact with each other in Radio Access Network (RAN). Virtualization of RAN functions is a way to solve some of the main challenges network operators face in deployment of new services, for example excessive time to market, increasing cost of energy, security and reliability.

MEC applications may contribute to increasing of user experience and efficient utilization of network resources [2]. MEC provides real-time network data such as radio conditions, network statistics, etc, for authorized applications to offer context-related services that can differentiate end user experience. MEC use cases and deployment options are presented in [5]. For reasons of performance, cost and scalability, MEC may be deployed at the radio node, at an aggregation point, or at the

edge of the core network. MEC A state-of-the-art research efforts and challenges on MEC domain are in presented in [6].

As to [7], in order to enable the development of viable MEC ecosystem, it is important to develop Application Programming Interfaces (APIs) that are simple as possible and are directly answering the needs of applications. To the extent this is possible, MEC needs to reuse existing APIs that fulfill the requirements.

MEC-service platform provides three types of middleware services: infrastructure services, radio network information services and traffic offload function [8]. Infrastructure services are used by applications for communications, service discovery and integration. Communication between MEC services and applications is based on Service-oriented Architecture (SOA). The applications access MEC services and other applications hosted on the same MEC platform through Web Services Application Programming Interfaces. Discovery and integration services provide visibility of services available on the MEC platform and enables flexible application deployment. Applications may discover service status, locate its end points and publish their own end point for usage of other applications. The access to MEC services is secured (authenticated and authorized).

Radio network information services provide authorized applications with low level real-time radio and network information related to users and cells. The traffic offload function prioritizes traffic and routes the selected, user-data stream to and from applications that are authorized to receive the data.

The security of Web Services and the XML based communications is of great importance to the overall security of distributed systems. Furthermore, in order to facilitate interoperability, the security mechanisms should preferably be based on established standards. In [9], the authors evaluate the importance of XML Signature and XML Encryption for *WSSecurity*. A review of Web Service security research in the field of cloud security is provided in [10]. While a lot of research is done on Web Service cryptography and digital signatures, there is a lack of works related to integrity of SOA-based systems.

In this paper, we present an approach to design infrastructure MEC Web Services for trust, security and load management. The proposed MEC Web Services allow application registration, discovery of available MEC services and registered applications, signing service agreement for MEC service usage, as well as control on internal load management. The paper is structured as follows. Section II presents the description of MEC Web Service that may be used for application authentication and registration, discovery of available MEC services and registered applications. Section III describes Web Service interfaces. Section IV presents in brief interfaces of MEC Web Service for internal load management. The conclusion summarizes the contributions.

2. Description of Application Registration Web Service

Generic requirements to MEC platform include management of the application lifecycle, provisioning of application environment which supports verification of application authenticity and integrity, as well as application mobility.

The complete cycle for using MEC application consists of three phases:

1. Authentication. Before using MEC services, the Application needs to be authenticated by the MEC service platform which prevents from unauthorized access. The authentication may be mutual e.g. the Application also authenticates the MEC platform,

2. Service selection. Once authenticated, the application selects service/application to be used. To ensure nonrepudiation, the MEC service platform can request a signing of a service agreement before allowing the service/application to be used.

3. Service use. Only after authentication, service selection and signing the service agreement have been done, the application can start using the actual service/application.

Apart from providing security, the authentication and service selection process allows MEC service platform provider to determine access right profiles for different applications. The amount of permissions can be made on the level of trust awarded to the application.

Figure 1 illustrates the sequence diagram for application authentication and MEC service discovery, selection and signing of

service agreement.

The procedure for initial access is as follows.

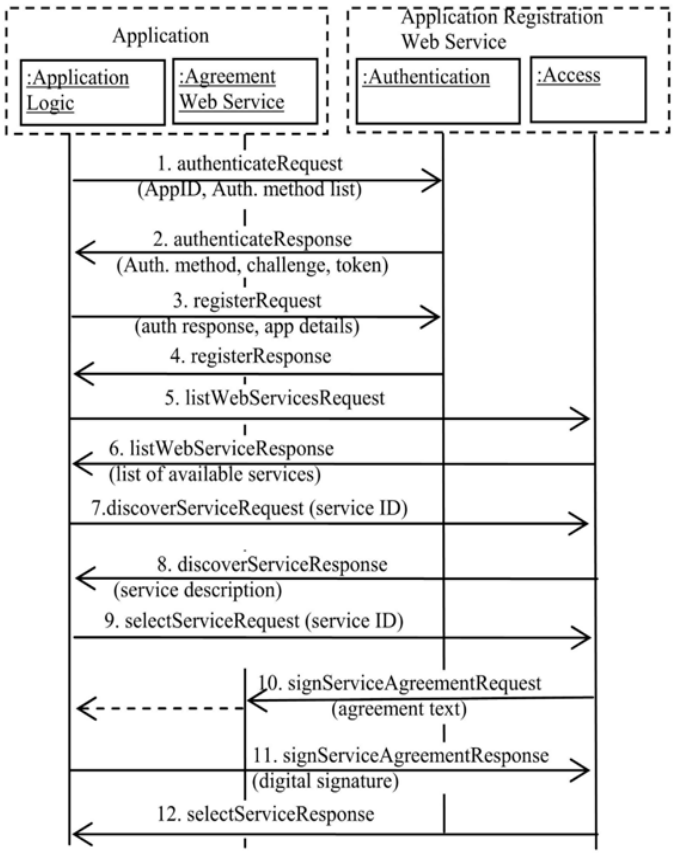

Figure 1. Illustrates the sequence diagram for application authentication and MEC service discovery, selection and signing of service agreement

1. At initial contact, the Application requests authentication by invoking authenticate operation. As the Application may support different authentication mechanisms, it provides as parameters the application identifier and list of the supported authentication mechanisms. The authentication mechanisms may be supported by cryptographic processes to provide confidentiality, and by digital signatures to ensure integrity. The inclusion of cryptographic processes and digital signatures in the authentication procedure depends on the type of selected authentication method. In some cases strong authentication may need to be enforced by the MEC platform to prevent misuse of resources.

2. The authentication between the Application and MEC platform is based on challenge-handshake authentication protocol. The Application and the MEC platform share a secret key. The MEC platform chooses an authentication method based on the capabilities of the Application. The MEC platform generates a challenge and based on the shared secret key computes a token and expected result. It responses to the Application with the chosen authentication method, the challenge and token.

3. The Application carries out a computation that resembles the generation of the token and the result at the MEC platform. If the calculated token is equal to the received one, the Application considers the MEC platform as authenticated. The Application invokes register operation and sends locally calculated result. The Application may send also some details that describe itself and may be used by other applications.

4. The MEC platform verifies whether the received result matches to the locally calculated one and if so, it sends to the

Application response for successful authentication and registration. If it does not, authentication fails.

5. When the MEC platform and the Application have successfully authenticated each other, the Application can request the list of existing MEC services by invoking *listWebServices* operation.

6. The MEC platform returns a list of existing Web Services with information about service ID and whether the respective Web Service is currently available or unavailable. Fig. 1. Application authentication and MEC service discovery, selection and signing of service agreement.

7. In order to discover the desired service, the Application may invoke *discoverService* operation which request more details about particular Web Service.

8. The MEC platform responses with a description of the Web Service.

9. The Application selects the Web Service by invoking *selectService* operation.

10. The MEC service requests that the Application confirms the intention to use the service by signing a service agreement. The signing of service agreement is to ensure nonrepudiation i.e. to prevent the Application from denying it has used the service. The service agreement may be done by digital signature.

11. The application signs the service agreement. Digital signature parameter contains cryptographic message syntax object (as defined in RFC 2630) with content type of signed data.

12. Once the service agreement has been successfully signed, the Application received an initial access to the Web Service and can start using it. The Application may subscribe for changes in the availability status of MEC service.

The Application starts notification with criteria defined. When the availability status of a MEC service changes, a notification message is sent to the Application. When the duration or count for notifications has been completed, the Application is notified.

Figure 2 shows a sequence diagram for notifications about Web Service availability status.

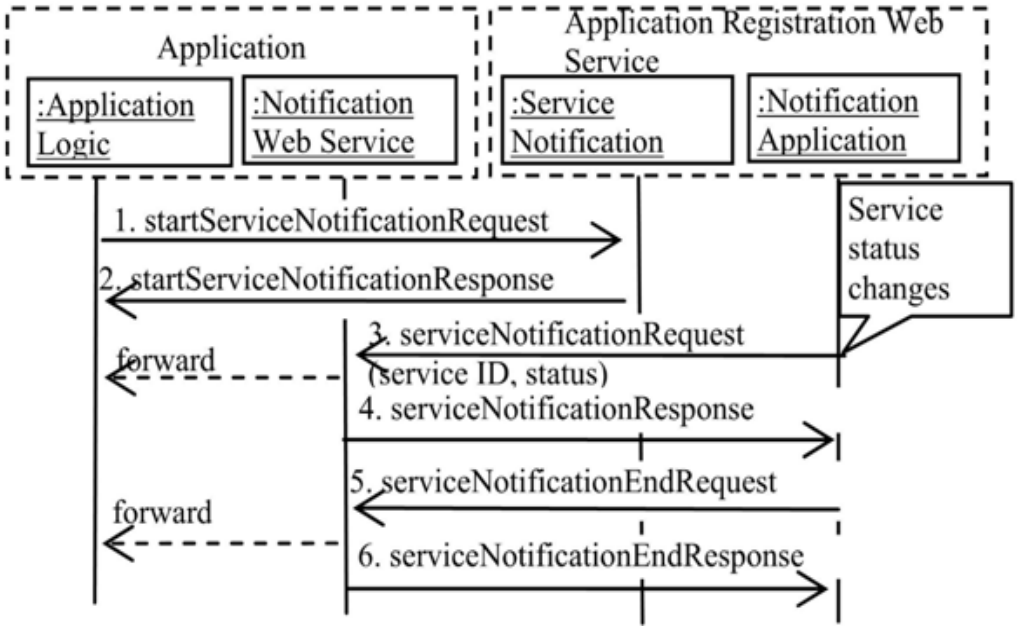

Figure 2 . Triggered notifications about changs in MEC Service availability status

3. Application Registration Interfaces

Application Registration Web Service provides interfaces for trust and security management. These interfaces support methods for application authentication, service/application, application selection and access to selected service/ application.

The Authentication interface is used for mutual authentication, application registration and deregistration. The authenticate operation is used by the Application to initiate authentication. The register operation is used by the Application to register with the MEC platform in case of successful authentication. The deregister operation is used by the Application to deregister from the MEC platform.

The Access interface is used by the application to obtain the identities of available services and applications and to sign service agreement for usage of selected service/applications. The Application uses *listServices* operation to request access to the identities of available services. The Application uses *listApplications* operation to request access to the identities of registered applications. With *discoverService* or *discoverApplication* operations the Application may receive more details about a particular Web Service or registered application. The Application uses *selectService* operation or *selectApplication* operation to select the MEC service or registered application.

The *ServiceAgreement* interface is used to sign an agreement that allows the Application to use the chosen service/application. It supports *signServiceAgreement* operation and *agreementTerminated* operation used to terminate the agreement. Once these contractual details have been agreed, the Application will be allowed to actually use it.
The Application Registration Web Service supports also interfaces that may be used to notify about MEC service or application availability.

The *ApplicationStatusNotificationManager* interface sets up notifications for the status of applications registered at the MEC platform. The *startApplicationNotification* operation is used to make available notifications about the registration status of applications. The stopApplicationNotification operation is used to terminate subscription for the registration status of applications. The interface to which notifications about application registration status are delivered is *ApplicationNotification*. It supports *applicationNotification* operation which notifies about changes in the registration status of an Application and *applicationError* which informs that notifications are cancelled. The *appNotificationEnd* operation informs the Application that the notifications have been completed when the duration or count for notifications have been completed.

The *ServiceStatusNotificationManager* interface sets up notifications for the status of MEC services. The *startServiceNotification* operation is used to make available notifications about the MEC services status. *The stopServiceNotification* operation is used to terminate notifications about the MEC service status.

The interface to which notifications about MEC service status are delivered is *ServiceNotification*. It supports *serviceNotification* operation which notifies about changes in the availability status of a MEC service and *serviceError* which informs that notifications are cancelled. The *serviceNotificationEnd* operation informs the Application that the notifications have been completed when the duration or count for notifications have been completed.

Figure 3 shows a simplified state diagram for Application registration and access to MEC services and applications.

In *ApplicationUnregistered* state, the Application is not registered. If the Application invokes authenticate operation, it provides its identity and supported authentication methods. In *MutualAuthentication* state, the MEC platform chooses an authentication method to be used, calculates a token and the expected result and challenges the Application. If the Application invokes register operation providing the right authentication response, the MEC framework registers the Application.

In *RegisteredApplication* state, the Application can request a list with available MEC service and discover a particular service. If the Application invokes s*electService* operation, the MEC platform initiates signing service agreement. In *ServiceAccessing* state, if the Application signs the service agreement, the MEC platform grants the service access. In *AccessGranted* state, the Application can request access to another MEC service or registered application. Before granting the access, the MEC platform requests signing service agreement.

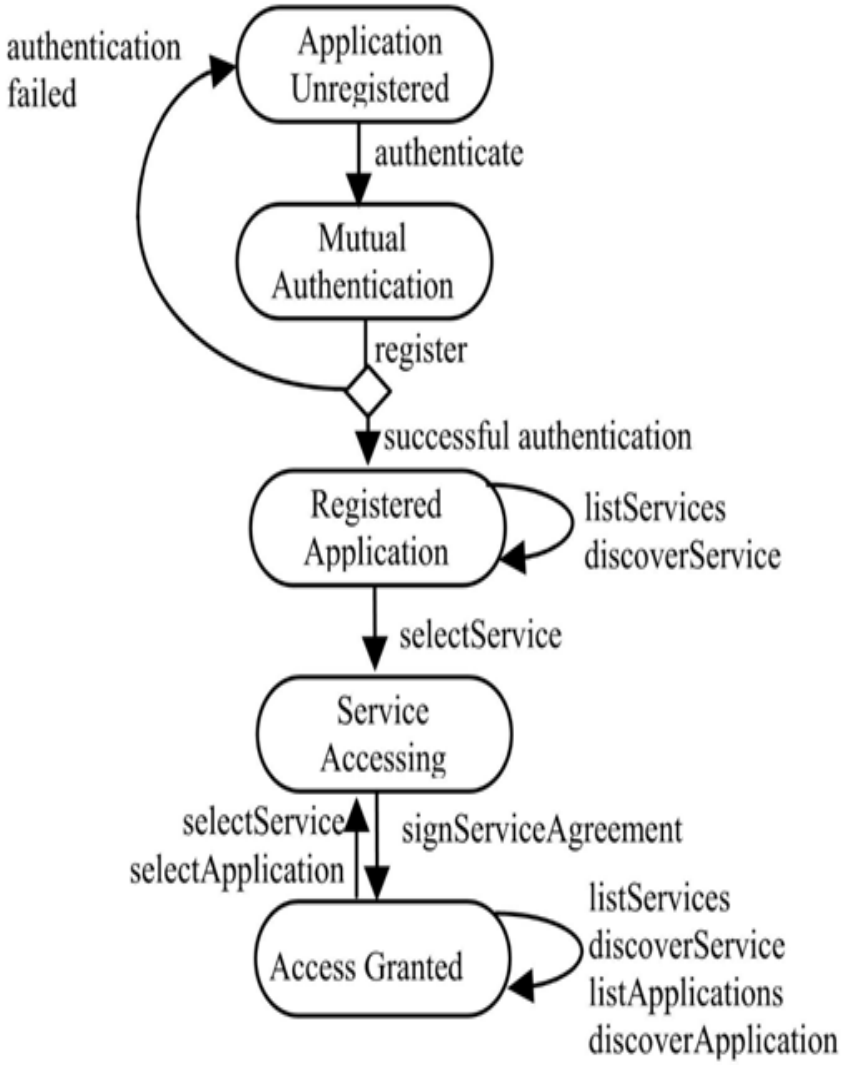

Figure 3. Registration and access to MEC services

4. Load Management Web Service

The Load Management Web Service allows the MEC platform to monitor and control the Application load according to a load management policy. The load management policy identifies rules for load management that the MEC platform needs to follow.

The *AppLoadManager* interface provides the following operations. The *startLoadNotification* operation is used by the MEC platform to subscribe for notifications about Application load level changes. The *stopLoadNotification* operation is used to terminate subscription for notifications about Application load level changes. The *holdNotification* operation is used by the MEC platform to hold temporary notifications from the Application. The MEC platform uses r*estoreNotification* operation to request the Application to restore sending notifications. The *appLoadStatisctic*s operation is used by the MEC platform to send to the Application statistics about its load level. The *appLoadStatError* operation is used by the MEC platform to inform the Application that load statistics can not be sent due to error.

The Load Manager interface provides the following operations. The Application uses *reportLoad* operation to report changes

in its load level. The *getAppLoadStatistics* operation is used by the Application to request from the MEC platform load statistics report.

Figure 4 shows a state diagram representing the MEC platform view of the Application and internal load levels.

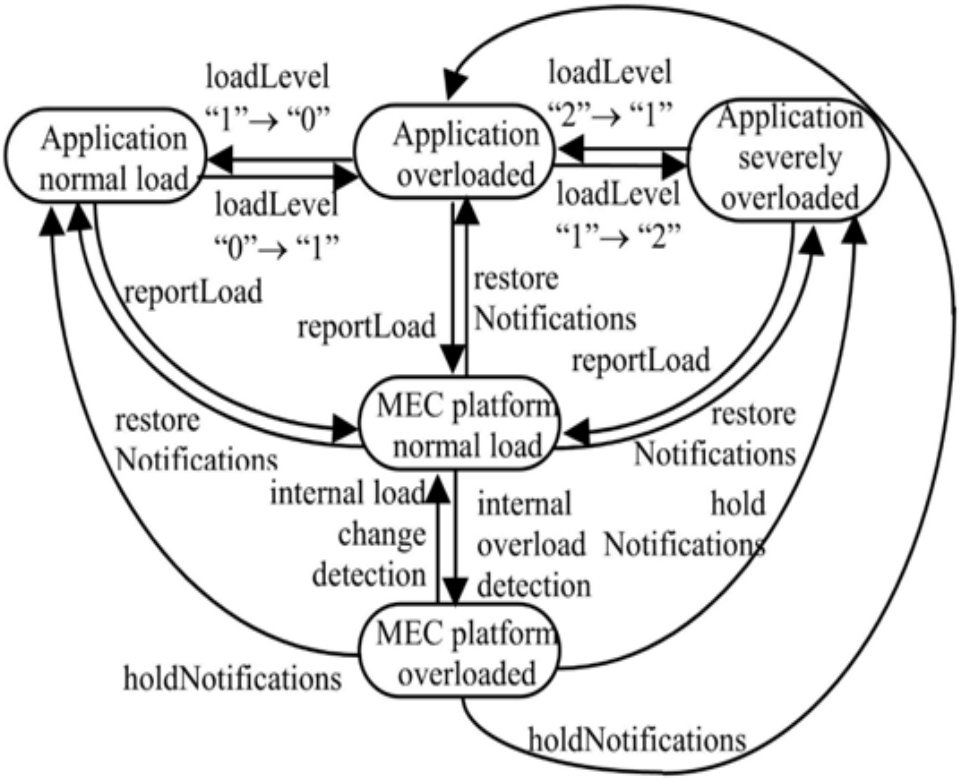

Figure 4. Application and MEC platform load level states

5. Conclusion

Mobile Edge Computing provides the ability to run IT based servers at the network edge, applying the concepts of cloud computing. A variety of new value added services can be provided with the integration of applications and radio network equipment. These services are oriented to improvement of quality of experience for mobile users, to enablement of disruptive Internet of Things services, to optimization of radio network performance, etc.

In this paper we propose an approach to design Infrastructure Web Services for MEC. Using the proposed MEC Web Services, applications can access MEC services and other applications hosted on the same MEC platform through Web Services Application Programming Interfaces. Trust and security management functionality provides applications with secured access to MEC service, visibility of available MEC services and other registered applications, thus enabling flexible application deployment. Applications may discover MEC service status as well as the status of other registered applications.

The proposed MEC infrastructure Web Services allow the network operator to manage the life cycle of the applications: deploy, start, stop and un-deploy more efficiently.

Acknowledgement

The research is conducted under the grant of project ÄH07/10-2016, funded by National Science Fund, Ministry of Education and Science, Bulgaria.

References

[1] Chen, Y., Ruchenbusch, L. (2016). Mobile Edge Computing: Brings the Value Back to Networks, IEEE Software Defined Networks, Newsletter, March 2016, Available at: http://sdn.ieee.org/newsletter/march-2016/mobile-edgecomputing-bring-the-values-back-to-networks

[2] Roman, R., Lopez, J., Mambo, M. (2016). Mobile Edge Computing, Fog et al.: A Survey and Analysis of Security Threats and Challenges", Future Generation Computer Systems, 2016, Available at: https://arxiv.org/pdf/1602.00484.pdf

[3] Beck, M. T., Feld, S., Linnhhoff-Popien, C., Pützschler, U. (2016). Mobile Edge Computing, Informatik-Spektrum, vol. 39, issue 2, pp. 108-114, 2016.

[4] Sarria, D., Park, D., Jo, M. (2016). Recovery for Overloaded Mobile Edge Computing, *Future Generation Computer Systems*, 2016, http://dx.doi.org/10.10.2016/j.future.2016.06.024

[5] Beck, M., Werner, M., Feld, S., Schimper, T. (2014). Mobile Edge Computing: A Taxonomy, 2014, The Sixth International Conference on Advances in Future Internet, IARIA, Available at:http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.670.9418&rep=rep1&type=pdf

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

[7] ETSI GS MEC 003, Mobile Edge Computing (MEC); Framework and Reference Architecture, v1.1.1, 2016. [8] ETSI GS MEC 002 Mobile Edge Computing (MEC); Technical Requirements, v1.1.1, 2016.

[9] Nordbotten, N. A. (2009). XML and Web Services Security Standards, *in IEEE Communications Surveys & Tutorials*, 11 (3), p 4-21, 3rd Quarter 2009.

[10] Cheong, C. P., Chatwin, C., Young, R. (2011). A New Secure Token for Enhancing Web Service Security, 2011 *IEEE International Conference on Computer Science and Automation Engineering*, Shanghai, 2011, p 45-48.