A Middleware Architecture for QoE Provisioning in Mobile Networks

Eirini Liotou1, Andrea Marotta2, Luigi Pomante2 and Kostas Ramantzas3

1 Department of Informatics & Telecommunications, National & Kapodistrian University of Athens, Greece, eliotou@di.uoa.gr  
2 Department of Information Engineering, Computer Science and Mathematics, University of L’ Aquila, Italy, luigi.pomante@univaq.it  
3 Iquadrat Informatica, Barcelona, Spain, kramantas@iquadrat.com

Abstract— This paper presents the design of the CASPER1 middleware architecture that enables the implementation of real-time QoE-driven service management. More in detail, we thoroughly describe the new modules, methods and functionalities that need to be integrated into the network providers’ infrastructure, according to CASPER. Based on this architecture, future virtualized networks can provide a closed-loop feedback environment that can act autonomously to maintain high levels of QoE with minimal human intervention. This is achieved via the proposed “QoE orchestrator”, which is responsible for implementing the QoE related logic.

Keywords—component; formatting; style; styling; insert (key words)

I. INTRODUCTION

The current paradigm in service provisioning lacks thorough end-to-end interpretation from the quality viewpoint, while the end-users'/customers’ profiles and preferences are mostly not taken into account. The subjective perception of a provided service, known as Quality of Experience (QoE) [1], is one of the most important factors for a user’s decision on retaining the service or giving it up, and the key parameter for enabling advanced Customer Experience Management (CEM). CASPER architecture exploits recent advances in communication networks, such as the Software Defined Networking (SDN) and the Network Functions Virtualization (NFV) to design and implement a middleware architecture for QoE-driven service provisioning. Therefore, the QoE provisioning logic in CASPER exploits the flexibility and re-configurability provided by the SDN/NFV technology in a bidirectional manner to acquire QoE influence factors from anchor points in the network and to apply QoE-driven service management policies. CASPER architecture is driven by three main objectives:

i) to design and optimise QoE estimation mechanisms for multimedia services;

ii) to design, and optimise QoE monitoring procedures;

iii) to design and optimise advanced QoE-driven service management policies.

To satisfy these objectives, CASPER architecture integrates the QoE management cycle (estimation-monitoring-management) [2]. It, therefore, consists of three interlinked modules, one-to-one mapped to these three instrumental functionalities required for the beneficial exploitation of QoE, i.e.:

i) reliable and passive QoE monitoring, including objective QoE estimation mechanisms;

ii) robust and real-time QoE-driven service management and CEM policies;

iii) efficient interfacing between the middleware and the network functions.

CASPER’s goal is that these three modules are optimized and released as an integrated solution, in order to accelerate the adoption of QoE-driven service provisioning in future networks. Moreover, another core requirement that defines the architecture design is the need to operate in real time (i.e., ~10 second intervals). The architecture is based on the following core concepts:

- Software Defined Networking;
- Virtualization;
- Big Data Analytics;
- Custom QoE intelligence.

The remainder of the paper is structured as follows: In sections II-IV we introduce fundamental elements and modules of the proposed architecture, while in sections V-VI we illustrate the QoE centered modules and functionalities. In section VII we describe the QoE management potential of the proposed architecture in terms of enabled use cases, while in section VIII we conclude the paper.

II. SDN-BASED MODULES AND FUNCTIONS

CASPER middleware architecture relies on SDN to configure network devices towards QoE-driven service provisioning. SDN can help provide a closed-loop feedback environment that acts autonomously to maintain QoE with minimal human intervention, providing real time control over an operator’s network. The two core SDN modules used in CASPER are OpenDaylight [3] and Open vSwitch [4].

CASPER employs OpenDaylight SDN controller, which provides a rich set of basic and extended network services for Network Resource Optimization (NRO). NRO is implemented in the form of flow rules, which are continuously pushed to the SDN switches via the OpenFlow protocol to optimize the diverse network applications (e.g., audio and video streaming, cloud applications, etc.) and fulfill the QoE constraints.

CASPER will employ virtual SDN switches in its proof of concept use cases. These will be implemented with Open vSwitch (OvS) software, a production quality, multilayer virtual switch licensed under the open source Apache 2.0 license. OvS communicates with the SDN controller via a Southbound interface, with the OpenFlow 1.3 protocol.
Typically, OvS instances run in fast x86 servers and each virtual switch port is mapped to an Ethernet port (Figure 1). In CASPER, we will experiment with low-cost ARM-based Single-Board Computers to implement SDN switches. To overcome the limitation of a single Ethernet port, which is the norm in SBC platforms, a VLAN (managed) switch will be employed, with five switch ports (one will be reserved as the trunk port that interconnects the SBC with the managed switch).

III. NFV-BASED MODULES AND FUNCTIONS

CASPER architecture follows the OPNFV architecture principles [5], although not restricted to exactly match them. The architecture, proposed in Figure 2, covers all NFV requirements as follows:

A. Hardware Infrastructure

An x86_64 Intel based server running CentOS Linux will be used for the underlying hardware running the NFV stack.

B. Virtualization Layer

KVM (Kernel-based Virtual Machine) is currently the industry’s dominant hypervisor for NFV deployments [6]. KVM will be therefore adopted as Virtual Computing platform for the CASPER project.

C. Virtual Network Functions (VNFs)

All VNFs will be running as KVM images, handled by OpenStack. VNFs should comprise the following:
- A disk image for instantiating the function;
- A compatible descriptor file, in order to register with the orchestrator.

D. Element Management System (EMS)

All VNFs deployed in the network should be accompanied with an EMS that provides APIs for access by their respective Virtual Function Manager. For the objectives of CASPER, ELK (Elasticsearch-Logstash-Kibana) will be acting as an element manager for gathering system and application logs from all running VNFs as well as the host server itself.

E. Virtualized Infrastructure Manager (VIM)

OpenStack is the industry’s unchallenged choice, so it will be used for this project.

F. VNF Manager(s)

Each VNF will provide its own configuration interface (either command line or web based). VNFs based on OpenDaylight natively have an EMS and VNF Manager as part of their core functionality.

G. Orchestrator

The orchestrator is a critical component as it is the interface between the QoE Manager and the underlying network. A major design decision is that the QoE Manager will act as an orchestrator, collecting data from ELK, and based on its internal rules will translate them to appropriate OpenStack and
OpenDaylight policies. These are then pushed via the respective Northbound APIs.

IV. ELK MODULE ANALYTICS

The ELK Stack is a collection of three open-source products - Elasticsearch, Logstash, and Kibana - from Elastic [7]. Elasticsearch is a Big Data distributed NoSQL database capable of handling huge data loads, while it can easily scale out just by adding new physical or virtual servers running the Elasticsearch engine. Logstash is a log pipeline tool that accepts inputs from various sources, executes different transformations, and exports the data to Elasticsearch. Kibana is a visualization layer that works on top of Elasticsearch.

Logstash will play the role of a simplified EMS in the project’s NFV architecture, while Kibana will provide real-time dashboards and visualizations that will be used for monitoring all experiments. Logstash will actively monitor all directories where VNFs generate logs as well as any SYSLOG servers, and as soon as a new log entry is discovered it will ingest it in Elasticsearch as a new JSON document.

As seen in Figure 3, Elasticsearch is also a source of data for the QoE Manager. The QoE Manager will be able to use JSON based queries over a Rest API to make near real-time complex queries about the performance of every network function.

V. QoE-CENTERED MODULES AND FUNCTIONS

The integration of QoE requirements into the CASPER architecture is presented in Figure 4. As presented in this figure, the QoE intelligence component acts as the NFV orchestrator in CASPER SDN/NFV architecture. It is therefore referred to as the “QoE orchestrator”. Three major functions comprise the QoE orchestrator, namely the QoE controller, QoE monitor and QoE manager. All three functions are integrated into the QoE orchestrator. QoE controller

The QoE controller is basically the function that controls the QoE monitoring process between the QoE orchestrator and the virtualized network. Through the envisioned interfaces of the QoE orchestrator, it allows a smooth flow of control and information.

The QoE controller materializes the JSON queries to the appropriate Element Management Systems (here, ELKs). These queries are driven by three major factors: a) the timing of the query or in other words the periodicity of the query, b) the nature of information to be queried or in other words the simple or synthetic Key Performance Indicators (KPIs) of interest (simple: e.g., number of packets transmitted, synthetic: e.g., packet loss ratio), and c) the appropriate ELK block (and VNF, thereof) that contains the queried information and needs to be addressed.

Each ELK is responsible for periodically collecting the appropriate information from the VNF that it is responsible for. For instance, information can be collected from various core (e.g., P-GW) or access network (e.g., eNB) VNFs that capture service quality, or from agents installed locally at end-devices that capture more subjective QoE influence factors (e.g., information from the UE application layer).

A. QoE monitor

The QoE monitor is in charge of estimating the QoE per flow using the Mean Opinion Score (MOS) scale or appropriate application-specific QoE metrics. A QoE estimation model is applied in order to measure the quality of the particular application at hand using MOS or application-specific KPIs. These QoE models are implemented by the Mobile Network Operator (MNO) inside the QoE monitor component of the QoE orchestrator. Therefore, these models can be easily updatable and manageable, as they constitute proprietary or standardized software functions.

B. QoE manager

The QoE manager is responsible for implementing QoE policies that increase the QoE of user flows, or improve the efficiency of the network, e.g. in terms of resource usage. Each management decision is a result of the input provided by the QoE controller, the quality estimations of the QoE monitor, and MNO-specific policies. The QoE management cycle, namely the way that QoE is controlled in the CASPER SDN/NFV architecture is presented in Figure 5.

QoE network-centered decisions can be implemented by the QoE manager using the capabilities offered either by OpenStack (which acts as the Virtualized Infrastructure Manager) or by OpenDaylight (which acts as the SDN controller and Virtual Networking Manager). For instance,
SDN is employed to improve QoE by enforcing policies via appropriate flow rules. Moreover, QoE constraints can influence caching in a Content Distribution Network (CDN), or control application-level parameters such as the coding rate of video streams.

VI. QoE MANAGEMENT POTENTIAL IN CASPER

CASPER architecture enables the realization and enforcement of QoE-based management schemes, towards improving the QoE of mobile users. The most characteristic QoE management techniques are presented next.

A. Dynamic Routing

This technique exploits the capabilities of OpenFlow. Open vSwitches are the central elements of interest in this case. The possibilities of flow handling include adding, updating and deleting flow entries, both reactively (in response to packets, after a problem appears) and proactively (before a quality degradation is perceived). The QoE orchestrator communicates the policies to the OpenDaylight controller regarding the handling of the flows, with the objective of maintaining or increasing the experienced quality of the users while they are communicating in real time. The dynamic routing of multiple flows of different applications and thus, different QoE requirements, is the major challenge in this case.

B. Adaptive instantiation of network functions and content

For achieving seamless service provisioning for mobile users, the NFV capabilities of CASPER architecture need to be exploited. Caching (or “CDN”) is considered as a virtual network function that can be manipulated by the QoE orchestrator, as depicted in Figure 4. The management logic in this use case is therefore that:

- Video content is moved (i.e., cached) closer to the end-user. This could be the case for a mobile user streaming video, where the location of the video content has a large impact on the streaming experience of the user.
- CDN servers are instantiated on demand. Instead of caching content to pre-determined server locations, another capability exposed through CASPER architecture is the instantiation of new video servers based on traffic demand. CDN servers are totally virtualized, and thus their management is allowed by the QoE orchestrator, with the objective of a) improving the quality of video segments downloaded and b) avoiding stalling events.
- Finally, the possibility of instantiating core network VNFs can have a large impact on QoE. For instance, shifting the S-/P-GW to a different hardware node that is closer to a region with temporarily and locally high network congestion is expected to significantly improve QoE of video streaming users.

C. Adaptive streaming control

Finally, CASPER architecture provides the potential to improve current adaptive video streaming schemes as follows:

- Monitor user buffers and ensure that stalling occurrences are minimized, if not diminished (e.g., by changing the routing of packets / prioritizing users with depleting buffers via scheduling or other mechanisms, etc.).
Monitor or even predict network congestion to provide end users with a more precise perception of the available bandwidth. In this way, users can take more informed decisions regarding the next segment requested to download.

Differentiate the QoE delivery according to the user subscription profile (e.g., prioritize the delivery of high-quality segments to premium users).

In the case that the MNO is the video provider itself, control the encoding of the video to better reflect the current or imminent network state.

VII. APIs AND INTERFACES

There are two types of APIs and interfaces that comprise CASPER, namely interfaces for connecting system components and interfaces for connecting application components. System components are items like Elasticsearch, QoE manager and OpenDaylight, while application components are VNFs like PGWs and Firewalls (e.g. Figure 6). Table 1 includes all APIs and interfaces as they will be implemented in the overall architecture.

![Figure 6: LTE related APIs](image)

VIII. CONCLUSION

This paper has presented the complete CASPER middleware architecture design. This architecture consists of SDN, NFV, Big Data and QoE-based modules and functions, while their in-between interfaces and interactions are identified and described. As a whole, the proposed CASPER architecture realizes a closed QoE management loop, with the potential to improve the end users’ QoE for diverse use cases and under different but challenging network conditions. Moreover, the architecture has been designed in a way that it ensures its smooth and realistic applicability to commercial products by taking into consideration real-life systems’ requirements, such as performance, complexity, and hardware.

ACKNOWLEDGMENT

This work has been funded by the EC under the auspices of the H2020-MSCA-RISE CASPER project (grant 645393).

REFERENCES

[1] Qualinet European Network on Quality of Experience in multimedia systems and services, “Definitions of Quality of Experience (QoE) and related concepts,” White Paper, 2012.

[2] E. Liotou, D. Tsolkas, N. Passas and L. Merakos, “Quality of Experience management in mobile cellular networks: Key issues and design challenges,” IEEE Communications Magazine, Network & Service Management Series, vol. 53, no. 7, July 2015.

[3] OpenDaylight - Open Source SDN Platform (A Linux Foundation Collaborative Project), online: https://www.opendaylight.org/platform-overview

[4] Open vSwitch (A Linux Foundation Collaborative Project), online: http://www.openvswitch.org/

[5] OPNFV Platform Overview, online: https://www.opnfv.org/software/technical-overview

[6] Kernel-based Virtual Machine, online: https://en.wikipedia.org/wiki/Kernel-based_Virtual_Machine

[7] Open Source Search & Analytics: Elasticsearch, online: https://www.elastic.co/

---

**TABLE I. KEY CASPER ARCHITECTURAL COMPONENTS**

| Component     | Function                        | Protocol | Direction | Used By          |
|---------------|----------------------------------|----------|-----------|------------------|
| OpenStack     | Compute, network, storage        | Rest     | Northbound| QoE Manager      |
|               | configuration                    |          |           | OpenDaylight     |
| Open vSwitch  | Provisioning, Management         | OpenFlow | Northbound| OpenDaylight     |
| OpenDaylight  | Provisioning, Management         | OpenFlow | Southbound| Open vSwitch     |
|               | Provisioning, Management         | Rest     | Northbound| OpenStack        |
|               |                                 |          |           | QoE Manager      |
| Elasticsearch | Data Ingestion Queries          | Rest JSON| Northbound| QoE Manager      |
| eNodeB        | S1-MME (Authentication)          | GTP      | Southbound| MME              |
| MME           | S1-MME (Authentication)          | GTP      | Northbound| eNodeB           |
| HSS           | S6a (Authentication)             | Diameter | Northbound| MME              |
| MME           | S6a (Authentication)             | Diameter | Southbound| HSS              |
| eNodeB        | S1-U (payload)                   | GTP      | Southbound| S-GW             |
| S-GW          | S1-U (payload)                   | GTP      | Northbound| eNodeB           |
| S-GW          | S5 (Payload local)               | GTP      | Southbound| P-GW             |
| P-GW          | S5 (Payload local)               | GTP      | Northbound| S-GW             |
| P-GW          | SGi                              | TCP/IP   | Southbound| Gi-LAN Applications (e.g., Firewall, IMS, Internet GW etc.) |