Timber: An SDN-based Emulation Platform for Experimental Research on Video Streaming

Arslan Ahmad, *Member, IEEE,* Alessandro Floris, *Member, IEEE,* and Luigi Atzori, *Senior Member, IEEE*

***Abstract*—In this paper, we present an open source Software-Defined Networking (SDN) based emulation platform called *Timber*. We aim to provide the research community with an experimental tool for the design and evaluation of the new Quality of Experience (QoE) management and monitoring procedures for video streaming. To this aim, the main functionalities of *Timber* include: i) an SDN application for taking QoE-aware management decisions; ii) an SDN controller to monitor the network's QoS (Quality of Service) and implement network management actions, such as network slicing and Multiprotocol Label Switching (MPLS) based prioritization operations; iii) a complete video streaming application including a multimedia server and a DASH-based client video player; iv) a user-end probe at the client video player to monitor QoE-related video application parameters, which are stored in a database that can be accessed by the SDN application; v) data analysis tools, which enable easy data visualization of measured QoS and QoE metrics as well as execution of statistical analysis of experimental results. In this article, we introduce and describe the main characteristics and functionalities of *Timber* as well as the implementation details. Finally, we provide experimental results of a video streaming scenario to demonstrate the capability of *Timber* to implement and test QoE-aware management approaches.**

***Index Terms*—Quality of Experience, QoE management, Video streaming, DASH, Software-Defined Networking (SDN).**

I. Introduction

THE enormous growth of multimedia traffic produced by Over The Top application providers (OTTs) and the increased quality expectations of end users have brought to the development of advanced network management approaches [1]. In particular, softwarization is expected to impact several aspects of network and application development by allowing different parties to instantiate and run a software-based architecture [2]. The aim is to offer a top-notch Quality of Service (QoS), which will be the base to provide a long lasting Quality of Experience (QoE) with a truly differentiated service provisioning on top of a shared underlying network infrastructure [3], [4].

One of the most relevant actors of network softwarization is the Software-Defined Networking (SDN) paradigm, which introduces programmability and open network access by decoupling the control plane from the data plane and via logically centralizing network intelligence. The SDN enables a new paradigm to the network and service management by the softwarization of the traffic engineering concepts together with information centric networking between application and network providers [5]. By keeping the global network view while performing network-wide management, the SDN based architectures and approaches for the QoE management of multimedia services have been widely proposed in the literature in the past decade [6], [7].

Simulation and emulation software are used for SDN implementation to mimic the environment of a real network and test it at minimum cost without the need for physical devices and real deployment. According to feature-based comparison conducted in [8], network emulation with Mininet is the most appropriate solution to middle scale networks whereas tools such as Mininet-CE, SDN Cloud DC, MaxiNet, and DOT are mostly suitable for large scale networks. On the other hand, network simulators (e.g., ns3) are limited mostly because external controllers are not supported. For this reason, network emulation environments are typically preferred. Among these, Mininet is the most used, which uses virtual hosts, switches and links to create a network, and uses the real network stack to process packets and connect to real networks. In addition, SDN control architecture tools, such as Ryu, Floodlight, OpenDayLight, are also available, which, in combination with Mininet and a media server (e.g., VLC) allow to implement and test network and application management approaches.

The mentioned emulation tools are well-designed to learn how to exploit at best the softwarization environment and test new algorithms for different networking protocols, to define novel QoS management strategies or to perform troubleshooting. However, in this scenario, there are not adequate tools to experiment with new solutions in the important field of QoE management for multimedia communications. Indeed, there is a need for an SDN-based platform which may assist the research community to implement and evaluate QoE management strategies and that should be hopefully open source. In particular, when considering QoE management there is a need to collect QoE-related information from client, network and application sides, and to feed this data to an appropriate QoE model. The measured QoE indicates the perceived service quality that drives service management actions, if needed [9].

For this reason, in this article, we introduce *Timber*, an open source SDN-based emulation platform for experimental research specifically designed for the QoE management of multimedia applications. In particular, we focused on video streaming applications as video is the most popular Internet content [10]. *Timber* covers all key stakeholders involved in the

The work on this paper was funded from the Regione Autonoma della Sardegna with POR–FESR funds (project EvIoT, RICERCA_1C–147) and from Italian Ministry of University and Research (MIUR), within the Smart Cities framework (Project Netergit, ID: PON04a200490).

A. Ahmad, A. Floris, and L. Atzori are with the Department of Electrical and Electronic Engineering, University of Cagliari, UdR CNIT of Cagliari, 09123, Cagliari, Italy (e-mail: arslan.ahmad.sh@gmail.com, alessandro.floris84@unica.it, l.atzori@ieee.org).

A preliminary version of this paper has been presented at QoMEX 2018.

video streaming service, i.e., client, network and application sides. *Timber* can perform the monitoring of both QoE-related parameters at the client side and QoS parameters at the network side. The measured information can then be used to take management actions on the network by implementing traffic engineering solutions or on the application by changing video streaming parameters. The main aim of *Timber* is to provide the research community with an open source and ready-to-use tool for an easy implementation and testing of new QoE-aware management procedures and techniques. Furthermore, *Timber* is totally programmed in Python, which is one of the most powerful and used programming languages nowadays.

Timber is developed on the top of the Mininet network emulator and Ryu SDN controller, the last one providing the major functionalities of the traffic engineering abstractions in the SDN environment. Moreover, *Timber* includes a complete end-to-end video streaming application composed of: i) a multimedia server for hosting and encoding the video contents to be suitable for the Dynamic Adaptive Streaming over HTTP (DASH) streaming technique; ii) a DASH-based client video player with probes to monitor QoE-related video application parameters, which are then stored in a cloud database (DB). The QoE-related information stored in the DB is accessible to the SDN application of *Timber* to take QoE-aware management decisions, which are translated into network management actions, such as network slicing and Multi-protocol Label Switching (MPLS) based prioritization operations, by the SDN network controller. *Timber* is also equipped with data analysis tools which allow to analyze and visualize the results of experimental researches without the need for additional tools.

The main contributions of this work and its importance in comparison with related works are:

- We introduce the major functionalities of *Timber* and how these can be used to experiment new solutions in the field of QoE management of video streaming applications;
- We provide a comparison among the most relevant studies using SDN-based platforms for QoE-aware management operations (operations aimed at optimizing end user's QoE by configuring service and network resources accordingly), and we highlight the differences concerning *Timber*;
- We made available *Timber* on GitHub[1] to allow other researchers to use it and improve it by adding new functionalities;
- We describe the main functionalities and implementation details of *Timber* including the complete video streaming application and the data analysis tools;
- We provide experimental results of a relevant video streaming scenario to prove the capability of *Timber* to implement and test QoE-aware collaborative network and application management approaches;

The paper is structured as follows: Section II presents the background studies regarding SDN-based service management approaches and adaptive video streaming services. Section III provides the motivations behind the implementation of *Timber* and its main characteristics. Section IV discusses the proposed SDN-based reference architecture of *Timber* and shows the description of the flow sequences among the different modules of *Timber*. Section V provides the *Timber* functionalities and implementation details while Section VI presents the experimental results to demonstrate the functionalities of *Timber*. Section VII concludes the paper.

[1]https://github.com/arslan-ahmad/Timber-DASH

II. Background

In this section, we discuss the state-of-the-art studies related to SDN-based service management approaches (Section II-A) and adaptive video streaming services (Section II-B).

A. SDN-based service management approaches

The SDN paradigm aims to simplify network management and to enable research innovations based on the decomposition of the control and data planes [5]. An SDN instance consists of three major components:

- *Application*: exploits the decoupled control and data plane to achieve specific goals. Applications communicate with a controller via the Northbound interface of the Control plane;
- *Control plane*: manipulates forwarding devices through a controller to achieve the specific goal of the application. The controller connects to the Data plane using the Southbound interface of the SDN-enabled switch;
- *Data plane*: supports a shared protocol (e.g., OpenFlow) with the controller and handles the actual packets based on the configurations that are manipulated by the controller.

In the following, some of the most relevant SDN-based network and application management approaches are surveyed from literature. Specifically, we considered the studies which implemented an SDN application where the QoE of the provided service is evaluated to drive QoE-aware network management actions via the SDN controller. Table I summarizes and compares these studies in terms of implementation tools, parameters monitored for QoE evaluation, implemented management actions and considered application.

The In-network QoE Measurement Framework (IQMF), proposed in [11], leverages on SDN to provide the flow duplication necessary for the measurement of QoE-related parameters. IQMF has been implemented and deployed in a large-scale SDN testbed to demonstrate its QoE measurement capabilities. Although with IQMF it is possible to monitor video quality-related metrics, network management functionalities have not been implemented. This work is extended in [12], where the UFair user-level application is introduced. The role of UFair is to orchestrate network resource allocation using SDN to mitigate QoE fluctuations and improve the overall QoE fairness in HTTP Adaptive Streaming (HAS) streaming. Precisely, the hierarchical fair resource allocation model slices a network into a hierarchy of zones and an aggregated utility is derived for each zone so that the resource provisioning can be conducted between zones, followed by the allocation within each zone. Optimization of QoE and QoE-fairness of HAS streaming is also the objective of the SDN-based approach presented in [13]. In this study, a set of connected application

TABLE I
COMPARISON OF SDN-BASED SERVICE MANAGEMENT APPROACHES PROPOSED IN THE LITERATURE.

Approach	Implementation tools	Monitored parameters	Management actions	Application
[11]	Dummynet, Floodlight controller, Scootplayer (DASH player), OpenCache	Video metrics: video quality and quality switches	Flow redirect, content placement and distribution (not implemented)	Video (HAS)
[12]	OpenStack, Ryu controller, Scootplayer (DASH player)	Fairness metrics: cost efficiency, switching impact and video quality	Network slicing into a hierarchy of zones	Video (HAS)
[13]	Mininet, Floodlight controller, Scootplayer (DASH player)	Video metrics: startup delay, number of stalling events, video quality, video quality switches, QoE-fairness	Path change, flow rate adaptation	Video (HAS)
[14]	OpenDayLight controller, VLC media player	QoS metric: bandwidth. Video metric: bitrate	Bandwidth allocation, video bitrate adaptation	Video (HAS)
[15]	Mininet, Ryu controller, HAS server, HAS player	QoS metrics and application parameters	Bandwidth allocation, video bitrate adaptation	Video (HAS)
[16]	Junos Space SDK, HTML5 video player	QoS metrics: TCP bandwidth, packet loss rate, jitter. Video metrics: player state and playout buffer status	Path change	Video (HAS)
[17]	Vienna simulator	Video metrics: stalling events, resolution, quality, bitrate, switching impact	Video segment encoding and placement (simulated)	Video (HAS)
[18]	Mininet, Open vSwitch, Floodlight controller, Iperf	QoS metrics: packet loss, throughput, latency, jitter	Flow rate adaptation, flow redirect, queue change	General
[19]	Mininet, OpenNetMon, OpenDaylight controller, D-ITG traffic generator, VLC media player	VoIP metrics: delay and packet loss. Video metrics: bitrate, frame rate and packet loss	Flow redirect	Video, VoIP
[20]	Mininet, Floodlight controller, Iperf, VLC media player, FTP server	QoS metrics: throughput, packet loss, delay	Path change, flow rate adaptation	Video, FTP
***Timber* (proposed)**	Mininet, Ryu controller, Python-based DASH client with user-end probe, DASH media server, data analysis & visualization tools	QoS metrics: throughput, packet loss, delay. Video metrics: buffer level, initial loading time, stalling duration, number of stalling events, playback bitrate and resolution, video quality	Network slicing, MPLS based flow prioritization	Video (DASH)

modules determine the appropriate quality adaptation and flow paths for HAS clients based on the network view provided by the SDN controller and on critical information collected from HAS servers. In [14], an SDN-based architecture is adopted for dynamic bandwidth allocation across multiple competing HAS streams. A video QoE optimization application (VQOA) is implemented to collect information from various points in the network by interacting with the SDN controller to provide a more accurate estimate of end-user QoE. The VQOA is also in charge of coordinating rate adaptation decisions in HAS clients and bandwidth allocation at the bottleneck link through the SDN controller to enhance end-user QoE. An end-to-end streaming architecture, called SDNHAS, is proposed in [15]. SDNHAS includes HAS players, HAS server and an SDN-based external application. This application selects the optimal bitrate and the corresponding quality that can maximize the viewer QoE while distributing the available bandwidth between active streaming sessions. A per-cluster decision algorithm (Q-learning based) is used to maximize QoE without any bandwidth violations and affecting other players QoE (QoE fairness optimization). The bitrate decisions provided by the application are forwarded to the SDN controller for proper bandwidth slicing and provisioning. In [16], a light-weight plugin implemented in the video player monitors various QoE factors (e.g., buffering status and video resolution) to analyze user-perceived experience when a video is playing. When buffering events occur, the SDN controller analyzes network conditions (e.g., TCP throughput and packet loss rate) on the streaming flow and changes the routing path in the network using MPLS traffic engineering. The QoE-SDN APP proposed in [17] focuses on mobile networks and aims to facilitate the feedback mechanism from the Mobile Network Operator (MNO) to the Video Service Provider (VSP), in order to enhance end-user QoE. Indeed, the QoE-SDN APP adopts common network and application programmability via the means of open APIs and can be flexibly programmed and customized to assure the desired QoE relying on the specifications of the SDN paradigm. QoE enhancement is achieved through proactive video selection and encoding to reduce the stalling probability. However, the primary focus of this study is on the application rather than on the complete SDN platform.

All the studies mentioned above focused on HAS services. However, other services may also be considered with SDN-based management approaches. The QoE App, presented in [18], maps measured QoS metrics into QoE in terms of the Mean Opinion Score (MOS). No specific media services are considered, and management actions are implemented to improve the QoS of general traffic flows intending to enhance the QoE of the services based on these flows. In [19], the SDN

QoE Monitoring Framework (SQMF) is an SDN application that monitors and preserves the QoE for Voice over IP (VoIP) and video applications. To ensure satisfactory QoE levels, various network parameters are periodically monitored and mapped into QoE using QoE estimation models; then, traffic redirection is implemented in case of low QoE detection on the transmission path. The SDN-based algorithm proposed in [20] aims to assure QoS and QoE of real-time multimedia applications by finding the optimal set of routes through the network, for each application, with the minimum flow cost subject to some constraints (e.g., available bandwidth, acceptable delay, and packet loss). The SDN controller provides continuous monitoring of the network status and allows to allocate new paths, if necessary, to avoid the link congestion effects, or at least sharply reduce them. Video streaming and FTP applications are considered for experiments.

Though some SDN-based QoE management solutions can be found in the literature, the software code of these solutions is not publicly provided impeding to implement and test novel functionalities and management algorithms on top of these solutions. Hence, there is a need for an open source ready-to-use SDN-based platform, which may assist the research community to evaluate QoE-aware management strategies for networks and applications. We aim to satisfy this need with *Timber*, our proposed SDN-based emulation platform.

B. Adaptive Video Streaming

In the last years, video streaming evolved from classical server-based to HAS. To adapt to dynamic network conditions, HAS requires the availability of small segments of the video encoded at different bitrates and quality levels at the server side. These are then selected by Adaptive Bitrate (ABR) algorithms running at the client side, which are typically based on monitored network state and player buffer occupancy [21]. The main objective of ABR algorithms is to maximise the video session quality by mostly trying to avoid stalling events which interrupt the video playback and are one of the most annoying factors impacting negatively the user's QoE [22]. This is achieved by selecting the segments encoded at the bitrate affordable by the network. For this reason, there are several examples of ABR algorithms implemented also from the industry side, such as Apple's HTTP Live Streaming[2] and Microsoft's Smooth Streaming[3].

The Moving Picture Experts Group (MPEG) has defined the standard MPEG-DASH (Dynamic Adaptive Streaming over HTTP) aimed at providing a system for uninterrupted video streaming service [23]. The DASH does not enforce any particular ABR algorithm, but it provides specifications regarding the encoding of the video in different representation rates and the description of video content information into the Media Presentation Description (MPD) files. The MPD file is requested by the client when the streaming session starts; video chunks are then sent from the HTTP server to the client as fast as possible to fill the playout buffer. The ABR algorithm comes into play once this buffer is full, by driving with its logic the download of new chunks to avoid stalling events. A comprehensive survey of rate adaptation techniques for DASH is provided in [22].

As video streaming is the most dominant application in the Internet, video service providers have to pay particular attention to the service quality perceived by the users. In [24], a tutorial on video quality assessment is presented. First assessment methods were solely based on QoS monitoring that did not consider the user's perceived QoE. On the contrary, subjective tests do reflect the perceived user's quality but cannot be conducted in real-time and are time consuming and costly. In the middle way, objective quality models identify and map the objective QoS parameters to QoE. The present and future assessment methods rely on data-driven models, which are based on machine learning (ML) algorithms trained with large-scale of data available for analyzing user QoE.

Although HAS outperforms server-side streaming by reducing the number of stalling events, actual implementations of QoE-driven solutions for video streaming are quite limited. Indeed, as discussed in [21], current HAS solutions only decide on ABR algorithms based on bandwidth measurements and buffer levels. As a result, resulting QoE is non optimal because it is influenced also by different types of factors (i.e., system, human and context [25]) that are not considered by these solutions. It is therefore needed to consider QoE models accounting for additional factors by involving all stakeholders, which in this case are the Internet Service Provider (ISP) and the OTT. With *Timber*, we aim to provide a tool able to collect information from all sides, i.e., client, network and application sides, so as to be able to implement and test QoE-aware management actions to improve the video streaming service.

III. Motivations and use cases

When designing an emulation platform mainly aimed at testing QoE-aware management approaches for multimedia services involving both network and application providers, the following functionalities should be considered [1]:

- *QoS monitoring*: responsible for the measurement of the state of the network by means of network QoS parameters, such as throughput, delay and packet loss;
- *QoS management*: network controller that applies traffic engineering actions aimed at decreasing the network load and optimizing traffic flow distribution;
- *Application management*: application controller that supports the optimization of application-related parameters;
- *QoE monitoring*: in charge of collecting QoE-related information (user-related parameters, network state and application state), which are the input of a mathematical model that measures the user's QoE;
- *QoE management*: QoE-based controller software that (on the basis of the current measured QoE, the network state and the application state) takes decisions regarding the correcting actions that should be taken on the network side, the application side, or both the network and application sides to optimize the end user's QoE.

It is evident that the QoE-aware management of multimedia services is complex, especially due to the different

[2]https://developer.apple.com/streaming/

[3]https://www.microsoft.com/silverlight/smoothstreaming/

information and functionalities in the hands of the two major actors, i.e., the network provider and the application provider. Indeed, whereas QoS monitoring and QoS management are in the hands of the network provider (although the application provider may monitor the QoS at the network edge and at the client side), the application provider may count on QoE monitoring and application management. With regard to the QoE management, both the providers may individually implement it but using different approaches: QoE-driven application management addresses control and adaptation on the end-user and application sides based on the available network resources; conversely, QoE-driven network management controls rely on performance and traffic monitoring solutions deployed within their access/core network without directly managing the applications.

However, past studies have shown that further potential and inherent benefits lie in integrated and cross-layer QoE management approaches, which include various forms of cooperative agreements and information exchange between involved stakeholders [7], [26]. It is therefore recommended that the QoE controller would be able to take decisions based on the combined information provided by both the providers to optimize the end users' QoE efficiently. However, technical issues, such as un-interoperability among different systems, may impede joint service management and discourage the providers to collaborate. For this reason, we based our emulation platform on the SDN paradigm, which makes use of virtualization and abstraction technologies to overcome the issues due to different system's hardware and to facilitate providers to share information to take combined controlling decisions aimed at service quality management.

With *Timber*, we aim to provide an SDN-based emulation platform that allows the research community to test the effectiveness of QoE-driven application and network management approaches as well as collaborative management approaches that need advanced tools for implementing information sharing and joint service controls. To this aim, the main functionalities of *Timber* are: i) an SDN application for taking QoE-aware management decisions; ii) an SDN controller to monitor the network's QoS and implement network management actions; iii) a multimedia server and a client video player to implement an end-to-end DASH video streaming service; iv) a user-end probe to monitor QoE related parameters, which are stored in a DB that can be accessed by the SDN application; v) data analysis tools, which enable easy data visualization of measured QoS and QoE metrics as well as execution of statistical analysis. *Timber* functionalities will be presented in detail in Section V.

Moreover, the main characteristics of *Timber*, which are summarized in Figure 1, are the following:

- *Open source*: *Timber* is completely implemented with open source software and is freely provided on GitHub as an open source platform[4].
- *Ease of customization*: as *Timber* is free and available as an open source platform, it can be used and customized by anyone with additional functionalities depending on the considered research activities. Additional code may regard, for example, client/server modules for different applications (e.g., VoIP, web browsing) or novel DASH client adaptation algorithms to be tested;
- *Plug & Play*: *Timber* is provided as a Virtual Machine (VM) including all its functionalities. This avoids users to install and configure every single component. However, the main components, such as the network topology and the client/server media modules, are fully configurable for the needed research experiments;
- *End-to-End service deployment*: *Timber* allows to deploy complete end-to-end services, from the media server to the client application passing through the transport network. It already implements an end-to-end DASH video streaming service;
- *QoE monitoring & management*: probes at the client side allow to monitor QoE-related parameters which are stored into a DB accessible by the SDN Application that can then implement QoE-aware management actions;
- *Python-based rapid prototyping*: all modules in *Timber* can be programmed by using the Python language, which supports rapid prototyping. Thus, the programming stack of *Timber* only depends on one programming language instead of multiple programming languages, which enables fast deployment of the experiments.

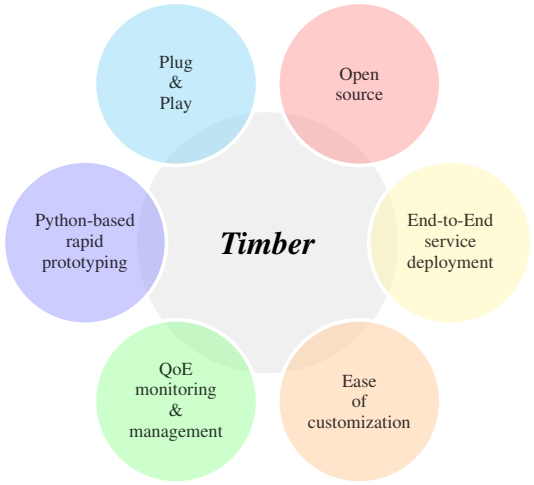

Fig. 1. Main characteristics of *Timber*.

IV. Timber Overview

This Section presents the SDN-based emulation platform, called *Timber*. In Section IV-A, we describe the reference architecture while Section IV-B explains the information flow sequence among the different modules of *Timber*.

A. Reference Architecture

Figure 2 shows the reference architecture representing two main actors of a typical multimedia communication scenario: the OTT application provider and the ISP. The architecture is composed of 5 planes. The top plane, i.e., the Application Management plane, concerns the cloud space owned by the

[4]https://github.com/arslan-ahmad/Timber-DASH

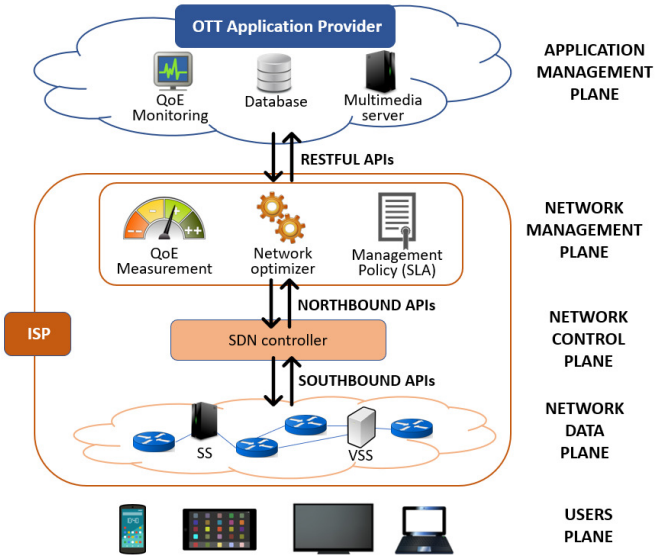

Fig. 2. Reference architecture.

OTT. The next three planes, namely, the Network Management plane, the Network Control plane and the Network Data plane, concern the ISP. At the bottom, there is the Users plane.

In the following, we provide the details for each plane.

1) Users plane: This plane considers the users of the multimedia application provided by the OTT through the ISP network. The OTT acquires QoE-related information from the probe installed at the user's device.

2) Network Data plane: This plane consists of the ISP network, which includes all physical and virtual network devices (e.g., switches, vswitches and routers) as well as Surrogate Servers (SSs) and Virtual Surrogate Servers (VSSs) for multimedia content replication. This plane is responsible for forwarding data traffic over the network according to commands provided by the Network Control plane through Southbound APIs.

3) Network Control plane: This plane includes the SDN controller through which network optimization decisions are implemented to the network. The SDN controller communicates with the Network Management plane via Northbound APIs (RESTful APIs) and implements the network optimizations in the network devices through Southbound APIs (OpenFlow protocol).

4) Network Management plane: This plane represents the application layer of the SDN architecture controlled by the network provider. This plane is responsible for taking network management decisions based on QoE measurement and network management policies. It includes: i) the QoE Measurement module, which measures the QoE delivered to the users (typically in terms of the MOS) based on a QoE model specific for the particular application. The QoE measurement module accesses the monitored application QoE Key Quality Indicators (KQIs) from the DB of the OTT via RESTful APIs. These KQIs, together with the network QoS measured by the SDN controller, are the input of the QoE model to measure the delivered QoE. Then, the measured QoE is provided as feedback to the Network Optimizer; ii) the Management Policy module, which takes into account the Service Level Agreements (SLA) between the user and the ISP and provides the feedback to the Network Optimizer; iii) the Network Optimizer, which receives the feedback provided by the QoE Measurement and Management Policy modules and runs the QoE-based network optimization algorithm. On the basis of the algorithm's objectives, this module decides which network control actions should be implemented by the SDN controller to optimize the network resources with the aim to improve the end user's QoE. Control actions are sent to the SDN controller via northbound APIs (RESTful APIs).

5) Application Management Plane: This plane considers the cloud space owned by the OTT. It includes: i) the QoE Monitoring module, which monitors the QoE KQIs of the delivered application using a probe installed at the user's device; ii) the DB, also accessible by the ISP, where the monitored KQIs are stored; iii) the Multimedia server, which contains the multimedia contents to be delivered to the users.

B. Information Flow Sequence

Figure 3 illustrates the two main flow sequences among the different modules of *Timber*: the Media Traffic flow and the Information flow. The former is the traffic flow of the multimedia content between the multimedia server and the DASH player. The latter concerns the information flows and includes the following steps:

1) At the start, the DASH client requests the MPD file from the multimedia server, which is followed by the server's reply with the MPD file.
2) The MPD file is parsed to the MPD reader.
3) The MPD reader reads the information regarding the available media representation such as representations' resolutions, number of segments, segment sizes, segment location on the media server, representation bitrates and total video length. The MPD reader provides feedback to the client adaptation algorithm module in the JSON format.
4) The client adaptation algorithm module provides the feedback to the DASH player for requesting the video segments based on the considered adaptation algorithm.
5) The DASH player requests the video segments from the multimedia server, which is followed by the reply from the server.
6) Based on the DASH player events, the KQIs, player stats, and buffer stats are collected by the logging module and user-end probe. The logging module creates the logs file in the local directory of the client player.
7) The user-end probe sends the collected information to the shared DB.
8) The QoE measurement module in the SDN controller application retrieves the QoE information from the DB via RESTful APIs.
9) The QoE measurement module computes the QoE based on the QoE model and provides output as feedback to the QoE management algorithm.

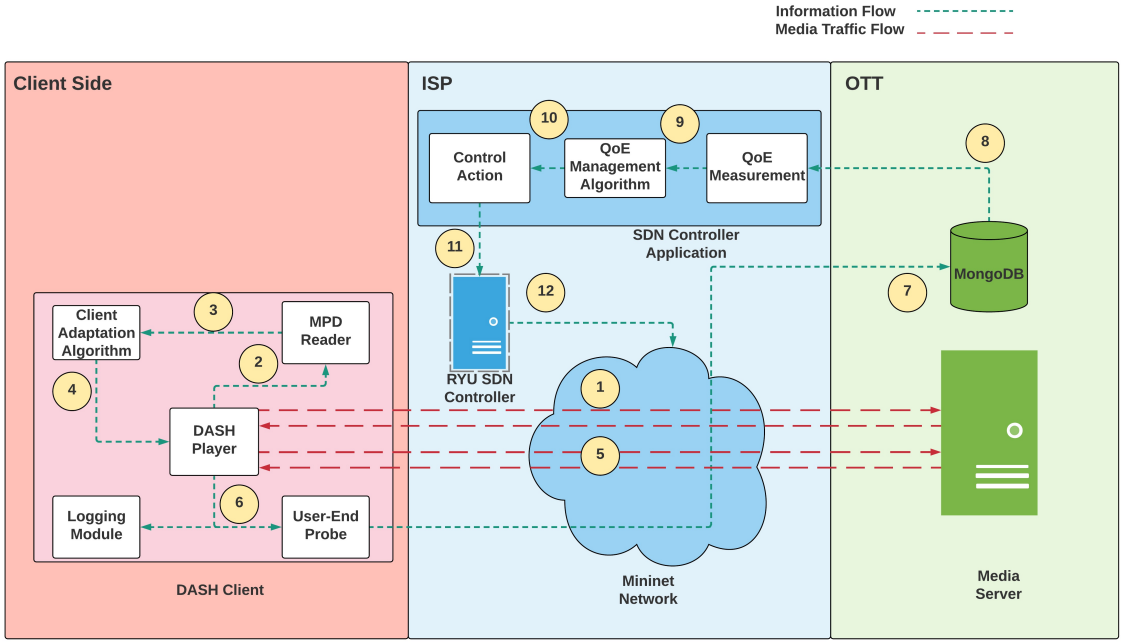

Fig. 3. Diagram of the information flow and media traffic flow of *Timber*.

10) The QoE management algorithm considers the measured QoE provided by the QoE measurement module and provides feedback to the control action module.
11) The control action module orchestrates the QoE management algorithm feedback to the SDN controller to perform the network control operations through RESTful APIs.
12) The SDN controller performs the network control operations based on the feedback provided by the control action module.

V. Timber Functionalities and Implementation

Timber functionalities are broadly categorized into five categories, which are summarized in Figure 4. To implement these functionalities *Timber* is based on several open source software packages, which are listed in Table II. In this section, we describe the implementation details for each functionality of *Timber*.

A. DASH Client

The DASH client supports the deployment of custom client-side adaptation algorithms, whose performance can be evaluated with *Timber*. The DASH client is implemented in Python language and consists of four modules: i) DASH player: the role of the DASH player is to initiate the communication with the multimedia server and play the video. In *Timber*, the DASH player is implemented with *AStream*[5], an open-source Python-based client DASH video streaming player proposed in [27]; ii) Client Adaptation Algorithm module: allows to implement and test DASH client-side algorithms. In *Timber*, three algorithms are already implemented, namely, Basic algorithm (video segments are selected based on the estimated throughput such that video playback bitrates are less than the estimated throughput), Netflix algorithm proposed in [28],

[5]https://github.com/pari685/AStream

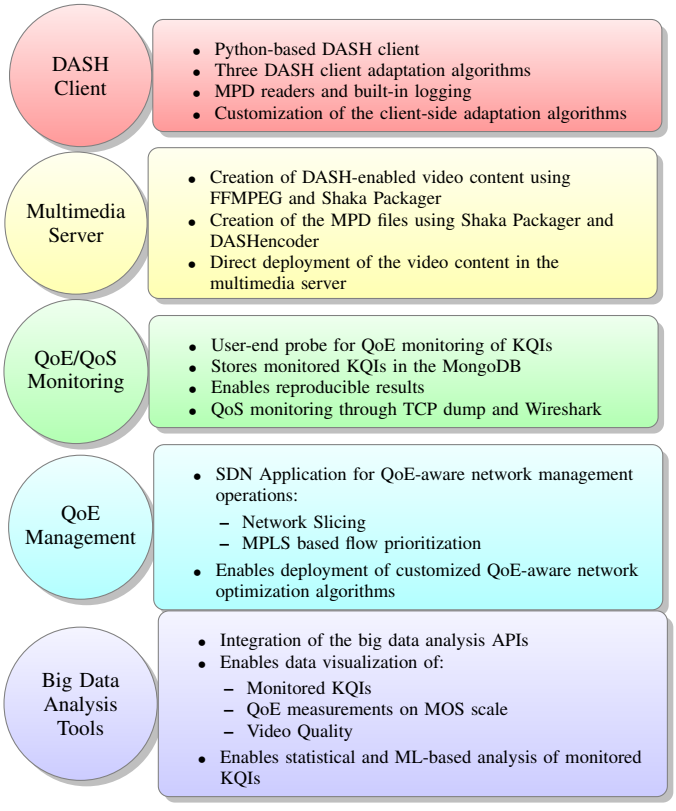

Fig. 4. Functionalities of *Timber*.

TABLE II
SOFTWARE PACKAGES USED IN THE CURRENT IMPLEMENTATION OF *Timber*.

Category	Software Package	Purpose
Client Side & QoE Monitoring	AStream	Client side implementation of the DASH video streaming application
	Mongo DB	Implementation of cloud DB for exchange of information between network and OTT
Virtual Network & SDN Controller	Mininet	Creation of virtual networks
	Ryu Controller	SDN controller for network management
Media Server	Apache HTTP Server	Server side implementation of content hosting server
	Shaka Packager	Creation of MPD files and DASH enable content preparation
	FFMPEG	Encoding the video content
Big Data Analytics	Jupyter Notebook, Pandas, Seaborn & Matplotlib	Data visualization
	Scipy, Numpy & Scikit-learn	Statistical & ML-based analysis

Segment Aware Rate Adaptation (SARA) presented in [27]; iii) MPD (Media Presentation Description) file reader: reads the MPD file (coded in XML) received from the multimedia server and extracts the information related to representations bitrate, resolution and video segment; iv) Logging Module: creates the log file (CSV format) based on the DASH player events in the local directory.

New client adaptation algorithms can be deployed by either modifying one of the aforementioned algorithms implementation file or by importing the Python based implementation of new client adaptation algorithm into the main DASH player module.

B. Multimedia Server

The multimedia server is implemented in the Ubuntu server 14.04 VM by Apache HTTP server[6], where MySQL server and PHP libraries for apache2 server are also installed as dependencies for the web/content hosting service. The video contents from the DB provided in [29] are hosted in the multimedia server VM, which also includes the FFMPEG tool for customized encoding of the contents. Shaka packager[7] and DASHencoder[8] are also installed to prepare the content according to the MPEG-DASH standard for DASH enabled video streaming [30]. Both Shaka Packager and DASHencoder create the video segments and MPD files for multiple resolutions of the encoded video content. For the experiments, the customized contents can be prepared for the DASH video streaming via Shaka Packager and Dashenconder.

Note that the current version of *Timber* only supports video streaming from the multimedia server (where DASH based video streaming files are stored) to the DASH client whereas live video streaming is not currently supported. Moreover, *Timber* can work in case the video sequence has to be rendered to improve the visual quality. However, more advanced rendering operations (e.g., rendering of 3D video) are not supported in the current implementation of *Timber*.

[6]https://httpd.apache.org
[7]https://github.com/google/shaka-packager
[8]https://github.com/slederer/DASHEncoder

C. QoE/QoS Monitoring

Timber can perform QoE monitoring through the user-end probe installed with the DASH client, which listens passively and collects video QoE-related application KQIs, namely, playout buffer level, player event stats (initial loading time, stalling duration, number of stalling events, timestamps of the player), and video streaming information (playback bitrate and resolutions). The monitored KQIs are sent and stored in the MongoDB database server, where they can be used for QoE measurement through means of mathematical QoE models.

The installation of the QoE monitoring probe at the client side also permits to collect information where the traffic is decrypted and passive monitoring probe can easily access application related KQIs. In this way, traffic encryption of real-world scenarios are avoided. This approach has been proposed and investigated in our previous works [31]–[33], where we also investigated the impact of the user-end probe frequency on the performance of the QoE management using SDN [34] and the impact of the user-end probe on the user device performance based on the device CPU consumption, RAM and battery utilization [31], [32]. From the study conducted in [34], it resulted that higher frequency of information exchange may result in better network reliability and delivered QoE, but a frequency higher than 1/4 Hz may not further improve the delivered QoE. Moreover, the studies in [31], [32] highlight that the event-based monitoring approach followed by the proposed user-end probe outperforms the traditional constant frequency based monitoring approach while utilizing less on-device resources.

Furthermore, the QoS monitoring is also supported by *Timber* via TCPdump and Wireshark through which QoS metrics, such as throughput, packet loss and delay, can also be acquired. These metrics can be used to monitor the state of the network and take network management actions accordingly.

D. QoE Management

Timber includes an SDN application that takes QoE-aware management decisions which are translated into network management operations by the SDN controller. For the creation of the virtual network environments, *Timber* is based on the Mininet emulator, which uses virtual hosts, switches and links

to create networks. The virtual host of the Mininet acts as a client by running the DASH client Python package. The SDN controller is implemented with the open Ryu SDN framework, which has been selected because it is an open SDN framework that provides software components with well-defined APIs that make it easy to create new network management and control applications. Moreover, Ryu supports various protocols for managing network devices, such as OpenFlow, Netconf, and OF-config. In particular, we were interested to OpenFlow support. Furthermore, Ryu is well-documented, continuously tested and written in Python (as all *Timber* components).

The Mininet Open vSwitches (OvS) are connected to the Ryu controller through the Southbound interface with the OpenFlow 1.3 protocol in the bridged configuration. The Ryu controller is used as the SDN controller to perform the network-wide monitoring and management tasks through the OpenFlow 1.3 protocol, which supports the following functionalities: 1) Configuration of OvSs through *ovs-vsctl* while keeping the global view of the network and; 2) Monitoring and administration of flow statistics and flow tables through *ovs-ofctl* respectively.

The SDN controller application runs on the top of the Ryu controller, which communicates through Northbound RESTful APIs. The controlling operations are divided into two categories based on traffic engineering concepts: 1) Network slicing and; 2) Flow prioritization by setting DSCP (Differentiated Services Code Point) prioritization classes and remarking of the flows (DiffServ). The SDN application is connected to the cloud DB (implemented by MongoDB) through the RESTful APIs which enable the SDN application to acquire KQIs (stored by the user-end probe) from the MongoDB through the secured HTTP GET response after the authentication. The retrieved information follows the JSON format, which is later sorted by the SDN controller application. The SDN controller application performs network level operations based on the retrieved information and supports the deployment of additional customized optimization algorithms.

The interested researchers can deploy their own algorithms for the QoE management on top of SDN controller by either modifying the already provided SDN applications in the platform or by developing the algorithm in the Python language and importing it into the Python program of SDN applications.

E. Big Data Analytics Tools

Timber contains a set of big data analytics tools based on Python, which supports deeper statistical and ML-based analysis of the experimental data stored in the DB. In particular, *Timber* big data analytics provides the statistical data visualization of the monitored KQIs, QoE measurement and video quality across the streaming session. Data visualization and analysis also support customization. The big data analytics tools are implemented by the integration of the Python language open source tools for data analysis. The Jupyter Notebook is installed with Pandas Python library in the platform to ease the customized analysis of the results. Python-based open source data visualization libraries, namely, Seaborn and Matplotlib are integrated to generate a customized graphical analysis of the data directly from the MongoDB database. Moreover, the Python library Plotly is also integrated for the interactive plots. Furthermore, for the more in-depth data analysis, the statistical and ML analysis tool kit is implemented through scipy, numpy, and scikit-learn libraries respectively. These libraries may assist the researchers to build/validate ML-based QoE prediction models from the data gathered from the experiments. Additionally, the ML algorithms can be easily deployed on top of SDN controller as applications by importing ML models via ML libraries. However, the training, testing and validation of the ML models is beyond the scope of this work as ML in itself is a wide research domain. Interested researchers can utilize the already provided Jupyter Notebook in the big data analytics tools for ML-based QoE prediction. The readers interested in more details regarding ML approach are referred to the Sklearn library [35].

With regard to the statistical analysis of the QoE measurements, we consider the subjective assessment based QoE model proposed in [36] for DASH video streaming, which is mathematically represented as Eq. (1):

$$MOS = 0.003 \cdot e^{0.064 \cdot t} + 2.498 \tag{1}$$

where t is the percentage of the video playback time that the client player spent on the highest layer during the video streaming session.

To measure the video quality[9] (VQ), we consider the VQ model proposed in [12], which not only has high correlation with the conducted subjective assessment but also comments on the VQ delivered to the users in different resolutions on the basis of the video encoding bitrates. Thus, the VQ delivered (on the scale (0-1)) to the users with the device resolutions $1080p$, $720p$ and $360p$ are computed using Eq.(2) to Eq.(4) respectively:

$$VQ_{1080} = -3.035 \cdot BR^{-0.5061} + 1.022 \tag{2}$$

$$VQ_{720} = -4.85 \cdot BR^{-0.647} + 1.011 \tag{3}$$

$$VQ_{360} = -17.53 \cdot BR^{-1.048} + 0.9912 \tag{4}$$

where BR represents the encoding bitrate of the video. Note that other models can be easily integrated in *Timber* as the platform provides the plug and play capability to deploy any QoE model for the QoE prediction. In order to deploy additional QoE models, Jupyter Notebook provided within the platform related to big data analytics can be modified. Moreover, to deploy other QoE model on top of SDN controller as an application, the QoE model can be easily added in the already provided QoE management application for network slicing and MPLS based flow prioritization.

VI. Experiments Setup and Scenarios

The main objective of the experiments is the demonstration of the major capabilities of *Timber* for the implementation and test of collaborative (involving collaborative actions between

[9]Note that video quality should not be mixed with QoE as video quality only represents the quality of the video while QoE is a broader term which includes not only the video quality but also other influencing factors related with the perceived quality by the user.

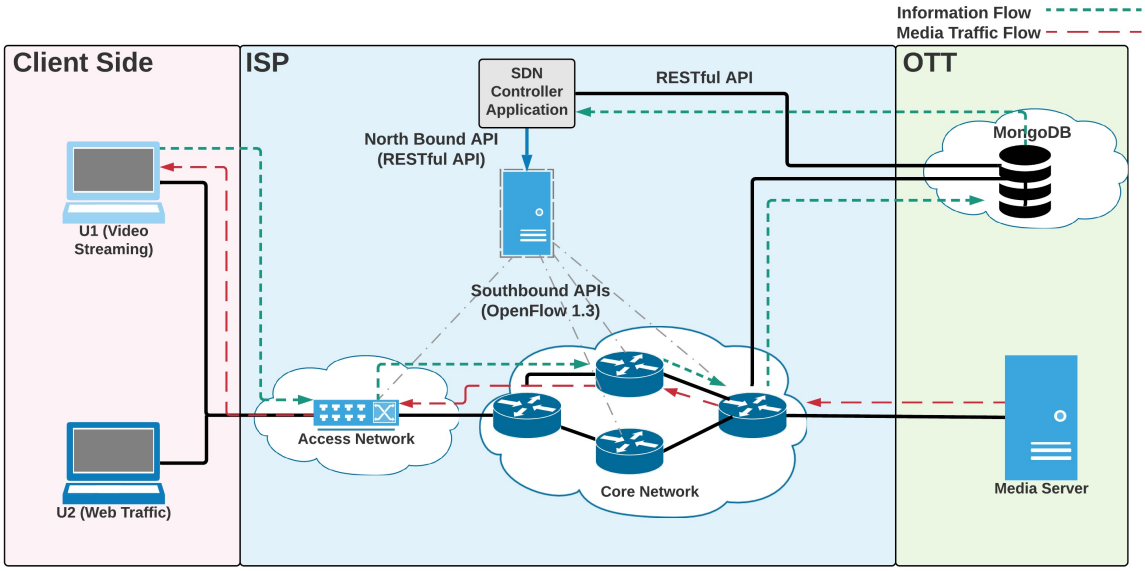

Fig. 5. *Timber* implementation details and Mininet topology used for the experiments.

the network and application providers) QoE-aware monitoring and management of video streaming service in the SDN paradigm. Additionally, the experimental results represent the *Timber* analytic capability for data visualization and statistical analysis. Note that *Timber* is not limited to only these experiments, and the focus of experiments is optimizing the QoE provisioning to video streaming. All the experiments follow the same topology illustrated in Figure 5 and regard the streaming of video sequences from the media server to User1 ($U1$) when web traffic is also transmitted on the same network towards User2 ($U2$). Indeed, $U1$ and $U2$ share the same access network.

Specifically, we considered the following two scenarios:

1) *No Collaboration (NC)*: no network optimization/traffic engineering concepts are applied via the SDN controller application, i.e., the network delivers the traffic on the best effort.
2) *Collaboration (C)*: QoE-related KQIs are exchanged throughout the active video session from client-side to SDN application via MongoDB. The SDN application performs the QoE measurement using the QoE model in Eq. (1). If the QoE measurement is lower than $MOS = 3$, the SDN controller application creates two virtual end-to-end network slices for the video streaming application and web traffic each of 4 Mbps. Then, the traffic flows generated by the video streaming and web traffic are assigned to the specific virtual network slice via flow matching criteria.

The experiments are conducted on the *Timber* platform that is built on two Linux Ubuntu VM servers 14.04 64 bits, i.e., VM-1 for the media server and VM-2 where client-side, Ryu SDN controller, Mininet and MongoDB are installed. Both VMs are installed on the Virtualbox with 3 GB RAM, 8 GB hard drive and core 2 Duo @2.4 GHz CPU resources.

The Big Buck Bunny 600 seconds long video sequence encoded with H.264 codec with 30 fps from the DB [29] is used in all experiments. The DASH video content has 4 seconds segment duration. Table III provides the details of the video representations used in the experiments. For all the experiments, the video streaming sessions are started in the $U1$ terminal via DASH client while the TCP protocol based web traffic of 6 Mbps is generated through the $U2$ terminal in all experiments settings. In order to create the network bottleneck, all links have fixed maximum bandwidth up to 8 Mbps, i.e., the HD video streaming requires $1.5-4$ Mbps in the selected video sequence case thus in the presence of the web traffic the network cannot guarantee HD video streaming. In all scenarios, the video sequence is played five times (5 runs).

Figure 6 represents the predicted QoE analysis of both scenarios. Figure 6-(a) provides the comparative analysis of the three client-side adaptation algorithms (Basic, Netflix, and SARA) in both scenarios in terms of the delivered QoE over the playback time. A significant improvement in the delivered QoE can be noticed in the Collaboration scenario where the controller performed the control actions based on the QoE measurement as compared to the NC scenario. Moreover, the highest delivered QoE is observed in the SARA client-adaptation algorithm while lower delivered QoE is noticed in the case of Netflix client-side adaptation algorithm. This is due to the conservative nature of the Netflix algorithm. Figure 6-(b) provides a comparison in terms of the aggregated predicted QoE. The box plot shows that for all the client-side adaptation algorithms, the Collaboration scenario delivers better QoE as compared to No collaboration. Moreover, the SARA client-side algorithm delivers the highest average QoE as compared to the Basic and Netflix algorithms in both scenarios. While the Netflix algorithm delivers the lowest QoE. Figure 6-(c) shows the histogram analysis of the delivered QoE where it can be noticed that SARA and basic algorithm perform better in both scenarios as compared to Netflix algorithm whereas a significant improvement in the delivered QoE can be

TABLE III
REPRESENTATIONS DETAILS OF THE VIDEOS USED IN THE EXPERIMENTS.

SD	**Resolution**	320x240			480x360					854x480	
	Bitrate (kbps)	45.226	88.783	128.503	177.437	217.761	255.865	323.047	378.355	509.09	577.751
HD	**Resolution**	1280x720			1920x1080						
	Bitrate (kbps)	1008.699	1207.152	1473.801	2409.742	2944.291	3340.509	3936.261			

observed in the Collaboration case. This highlights the *Timber* capability to deploy and provide a comparative analysis of client-side adaptation algorithms in combination with QoE-aware management algorithms.

Figure 7 provides a comparative analysis of the video quality (VQ). Note that VQ is computed as explained in Section V-E with Equations (2) to (4). Figure 7-(a) shows the VQ delivered over the playback time. In the case of NC, many VQ switching can be noticed which have a negative impact on the perceived QoE by the user. Whereas in case of Collaboration less VQ switching is observed. Among the client-side adaptation algorithms, highest VQ switching can be observed for the Basic algorithm while the Netflix algorithm demonstrates less VQ switch for both scenarios. Figure 7-(b) represents the aggregated VQ comparison where improvement can be noticed in the Collaboration case where QoE management algorithm performs network slicing. Moreover, the SARA algorithm delivers better average VQ as compared to basic and Netflix. Figure 7-(c) represents the histogram of the delivered VQ. A significant improvement can be observed in Basic algorithm from NC to Collaboration scenario whereas Netflix depicts a slight change in the VQ in both scenarios. SARA algorithm delivers better VQ in both cases. This demonstrates that *Timber* can perform QoE management and deployment of the client-side adaptation algorithms as well as a comparative analysis of the QoE management algorithms and client-side adaptation algorithms in terms of VQ.

Finally, Figure 8 shows the analysis of the playback bitrates for each scenario. Figure 8-(a) shows the video playback bitrate over playback time. It can be observed that the playback bitrate in the Collaboration case is higher as compared to the NC case. Moreover, SARA algorithm has the highest playback bitrates in both the scenarios. Figure 8-(b) shows the aggregated boxplot analysis of the bitrate playback where SARA algorithm demonstrates the highest average playback bitrate for both scenarios. Figure 8-(c) shows histogram analysis of the playback bitrate where significant improvement in all client adaptation algorithms can be noticed in case of Collaboration as compared to NC.

VII. CONCLUSION

We presented an open source SDN based emulation platform called *Timber* aimed to assist the research community to implement and evaluate QoE management strategies. The first part of the article describes the main characteristics and functionalities which distinguish *Timber* from state-of-the-art SDN based platforms. The most critical novelties are the free provision of *Timber* on GitHub and the easiness of installation through ready-to-use VMs as well as the integration of data analysis tools for data visualization and statistical analysis of experimental results.

In the second part, the implementation details of *Timber* are presented. Moreover, experimental results of a DASH-based video streaming scenario are provided, which prove the capability of *Timber* to implement QoE-aware management approaches which also include collaborative actions between the network and the application providers. Thanks to the ease of customization of *Timber*, future work will be focused on the implementation of additional modules which may enable different applications to be tested on *Timber*, such as VoIP and Web browsing. Moreover, novel DASH adaptation algorithms may be tested in the current version of the *Timber* platform.

REFERENCES

[1] 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*, vol. 53, no. 7, pp. 145–153, July 2015.

[2] A. A. Barakabitze, A. Ahmad, R. Mijumbi, and A. Hines, "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges," *Computer Networks*, vol. 167, p. 106984, 2020.

[3] A. Cuadra-Sánchez, M. Cutanda-Rodríguez, I. Pérez-Mateos, A. Aurelius, K. Brunnström, J. Laulajainen, M. Varela, and J. E. López De Vergara, "A global customer experience management architecture," in *2012 Future Network Mobile Summit (FutureNetw)*, July 2012, pp. 1–8.

[4] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network Slicing and Softwarization: A Survey on Principles, Enabling Technologies, and Solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[5] H. Farhady, H. Lee, and A. Nakao, "Software-Defined Networking: A survey," *Computer Networks*, vol. 81, pp. 79 – 95, 2015.

[6] A. A. Barakabitze, N. Barman, A. Ahmad, S. Zadtootaghaj, L. Sun, M. G. Martini, and L. Atzori, "QoE management of multimedia streaming services in future networks: a tutorial and survey," *IEEE Communications Surveys & Tutorials*, 2019.

[7] L. Skorin-Kapov, M. Varela, T. Hoßfeld, and K.-T. Chen, "A Survey of Emerging Concepts and Challenges for QoE Management of Multimedia Services," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, no. 2s, pp. 29:1–29:29, May 2018.

[8] S. Zavrak and M. Iskefiyeli, "A Feature-Based Comparison of SDN Emulation and Simulation Tools," in *International Conference on Engineering Technologies (ICENTE'17)*, 2017.

[9] A. Ahmad and L. Atzori, "MNO-OTT Collaborative Video Streaming in 5G: The Zero-rated QoE Approach for Quality and Resource Management," *IEEE Transactions on Network and Service Management*, 2019.

[10] Cisco. (2020) Cisco visual networking index: Forecast and trends, 2017–2022 white paper. [Online]. Available: https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-741490.html

[11] A. Farshad, P. Georgopoulos, M. Broadbent, M. Mu, and N. Race, "Leveraging SDN to Provide an In-network QoE Measurement Framework," in *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 2015, pp. 239–244.

[12] M. Mu, M. Broadbent, A. Farshad, N. Hart, D. Hutchison, Q. Ni, and N. Race, "A Scalable User Fairness Model for Adaptive Video Streaming Over SDN-Assisted Future Networks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 8, pp. 2168–2184, 2016.

[13] A. Erfanian, F. Tashtarian, and M. H. Yaghmaee, "On Maximizing QoE in AVC-Based HTTP Adaptive Streaming: An SDN Approach," in *2018 IEEE/ACM 26th Int. Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–10.

[14] S. Ramakrishnan, X. Zhu, F. Chan, and K. Kambhatla, "SDN Based QoE Optimization for HTTP-Based Adaptive Video Streaming," in *IEEE Int. Symposium on Multimedia (ISM), 2015*. IEEE, 2015, pp. 120–123.

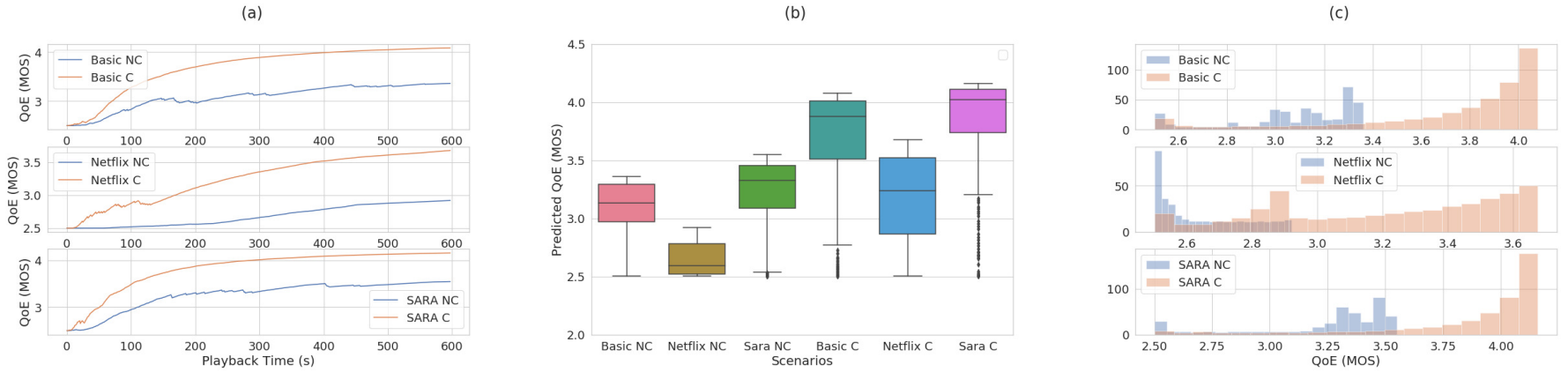

Fig. 6. Predicted QoE analysis: (a) Average predicted QoE over playback time; (b) Aggregated predicted QoE boxplot; (c) Histogram plot of the predicted QoE.

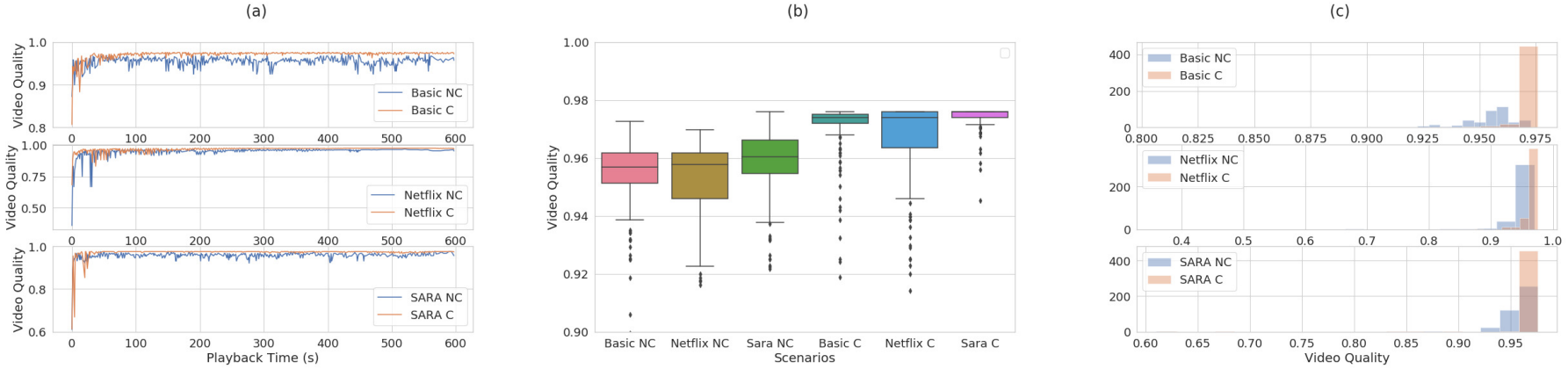

Fig. 7. Video quality analysis: (a) Average video quality over playback time; (b) Aggregated video quality boxplot; (c) Histogram plot of the video quality.

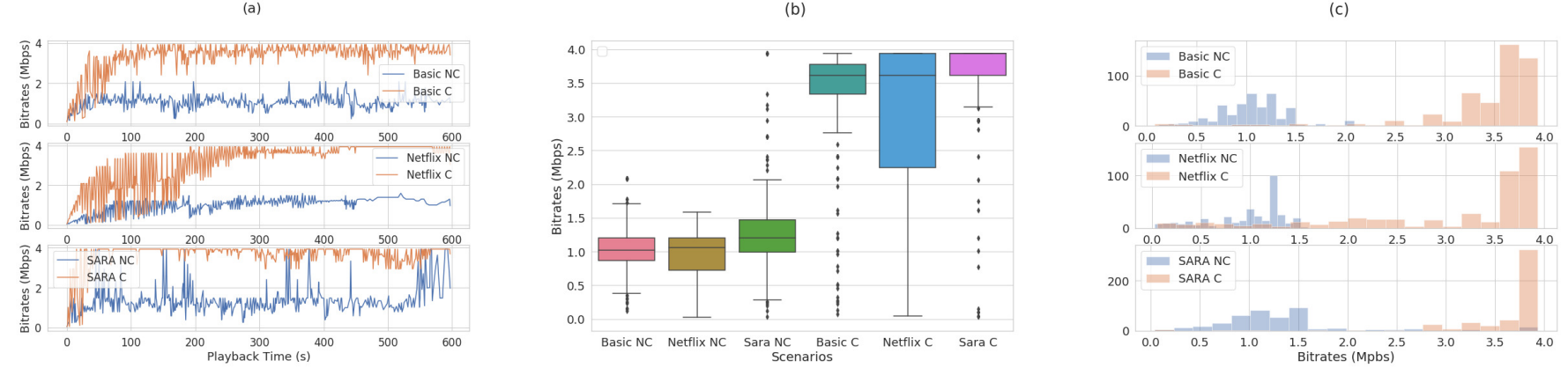

Fig. 8. Playback Bitrate analysis: (a) Average playback bitrate over playback time; (b) Aggregated playback bitrate boxplot; (c) Histogram plot of playback bitrate.

[15] A. Bentaleb, A. C. Begen, R. Zimmermann, and S. Harous, "SDNHAS: An SDN-Enabled Architecture to Optimize QoE in HTTP Adaptive Streaming," *IEEE Transactions on Multimedia*, vol. 19, no. 10, pp. 2136–2151, 2017.

[16] H. Nam, K.-H. Kim, J. Y. Kim, and H. Schulzrinne, "Towards QoE-aware video streaming using SDN," in *IEEE Global Communications Conference (GLOBECOM), 2014*. IEEE, 2014, pp. 1317–1322.

[17] E. Liotou, K. Samdanis, E. Pateromichelakis, N. Passas, and L. Merakos, "QoE-SDN APP: A Rate-guided QoE-aware SDN-APP for HTTP Adaptive Video Streaming," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 598–615, 2018.

[18] F. Volpato, M. P. D. Silva, A. L. Gonçalves, and M. A. R. Dantas, "An Autonomic QoE-Aware Management Architecture for Software-Defined Networking," in *2017 IEEE 26th Int. Conf. on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, 2017, pp. 220–225.

[19] M. Xezonaki, E. Liotou, N. Passas, and L. Merakos, "An SDN QoE Monitoring Framework for VoIP and Video Applications," in *2018 IEEE 19th Int. Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2018, pp. 1–6.

[20] F. Ongaro, E. Cerqueira, L. Foschini, A. Corradi, and M. Gerla, "Enhancing the quality level support for real-time multimedia applications in software-defined networks," in *2015 Int. Conf. on Computing, Networking and Communications (ICNC)*, 2015, pp. 505–509.

[21] M. Seufert, S. Egger, M. Slanina, T. Zinner, T. Hoßfeld, and P. Tran-Gia, "A Survey on Quality of Experience of HTTP Adaptive Streaming," *IEEE Communications Surveys Tutorials*, vol. 17, no. 1, pp. 469–492, Firstquarter 2015.

[22] J. Kua, G. Armitage, and P. Branch, "A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming Over HTTP," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1842–1866, thirdquarter 2017.

[23] MPEG. (2019) Standard ISO/IEC 23009-1:2019, Information technology - Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats. [Online]. Available: https://www.iso.org/standard/75485.html

[24] Y. Chen, K. Wu, and Q. Zhang, "From QoS to QoE: A Tutorial on Video Quality Assessment," *IEEE Communications Surveys Tutorials*, vol. 17, no. 2, pp. 1126–1165, Secondquarter 2015.

[25] P. Le Callet, S. Möller, A. Perkis *et al.*, "Qualinet White Paper on Definitions of Quality of Experience (2012)," in *European Network on Quality of Experience in Multimedia Systems and Services (COST Action IC 1003)*, Lausanne, Switzerland, Version 1.2, March 2013.

[26] A. Floris, A. Ahmad, and L. Atzori, "QoE-Aware OTT-ISP Collaboration in Service Management: Architecture and Approaches," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 14, no. 2s, pp. 36:1–36:24, Apr. 2018.

[27] P. Juluri, V. Tamarapalli, and D. Medhi, "Sara: Segment aware rate adaptation algorithm for dynamic adaptive streaming over http," in *2015 IEEE International Conference on Communication Workshop (ICCW)*. IEEE, 2015, pp. 1765–1770.

[28] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4, pp. 187–198, 2015.

[29] S. Lederer, C. Müller, and C. Timmerer, "Dynamic adaptive streaming over HTTP dataset," in *Proceedings of the 3rd Multimedia Systems Conference*. ACM, 2012, pp. 89–94.

[30] I. Sodagar, "The MPEG-DASH Standard for Multimedia Streaming Over the Internet," *IEEE MultiMedia*, vol. 18, no. 4, pp. 62–67, April 2011.

[31] A. Ahmad, L. Atzori, and M. G. Martini, "Qualia: A multilayer solution for QoE passive monitoring at the user terminal," in *IEEE Int. Conf. on Communications (ICC), 2017*. IEEE, 2017, pp. 1–6.

[32] A. Ahmad, A. Floris, and L. Atzori, "Towards QoE monitoring at user terminal: A monitoring approach based on quality degradation," in *2017 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*. IEEE, 2017, pp. 1–6.

[33] F. Wamser, M. Seufert, P. Casas, R. Irmer, P. Tran-Gia, and R. Schatz, "yomoapp: A tool for analyzing qoe of youtube http adaptive streaming in mobile networks," in *2015 European Conference on Networks and Communications (EuCNC)*.

[34] A. Ahmad, A. Floris, and L. Atzori, "Towards information-centric collaborative QoE management using SDN," in *2019 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2019, pp. 1–6.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] T. Hoßfeld, M. Seufert, C. Sieber, and T. Zinner, "Assessing effect sizes of influence factors towards a QoE model for HTTP adaptive streaming," in *Quality of Multimedia Experience (QoMEX), 2014 Sixth International Workshop on*. IEEE, 2014, pp. 111–116.

Arslan Ahmad is a Ph.D. in Electronics and Computer Engineering. He received his engineering degree in Aviation Electronics (Avionics Engineering) in 2011 from PAF College of Aeronautical Engineering, National University of Sciences and Technology, Pakistan. In 2014, he received Master's degree in Computational Sciences and Engineering from Research Center for Modeling & Simulation, National University of Sciences and Technology, Pakistan. He received his Ph.D. degree in Electronics and Computer Engineering from the Department of Electrical and Electronics Engineering, University of Cagliari, Italy in 2019. He has worked as Marie Curie Fellow (2015-2018) in MSCA ITN QoE-Net. He has served as a Postdoctoral researcher at University of Cagliari from 2018 to 2019. He has numerous publications in International peer-reviewed conferences and journals, two of which has been awarded Best paper award at IFIP/IEEE IM 2017 and IEEE Multimedia Technical Committee. He has been the reviewer of IEEE Transaction on Multimedia, Springer Multimedia Tools and Application, and Elsevier Image Communication. He has been Web Chair of the 10th International Conference on Quality of Multimedia Experience (QoMEX 2018) and Technical Program Committee member of IEEE ICC and IEEE WCNC. His research interests include multimedia communication, Software-Defined Networking (SDN), Network Function Virtualization (NFV), machine learning, Quality of Experience (QoE) based service and network monitoring and management. The primary focus of his research is the QoE monitoring and management of the Over-The-Top (OTTs) multimedia services and OTT-ISP collaboration for the joint multimedia service management.

Alessandro Floris is an Assistant Professor at the Department of Electrical and Electronic Engineering, University of Cagliari (Italy). He received his M. Sc. degree in Electronic Engineering and his PhD in Electronic and Computer Engineering from University of Cagliari respectively in 2011 and 2017. He is a member of the Multimedia & Communications Laboratory (MCLab) (mclab.diee.unica.it.) and of the Italian University Consortium for Telecommunications (CNIT). His main research interests include multimedia applications, Quality of Experience (QoE) based service and network management systems, Internet of Things (IoT) applications and Smart Cities.

Luigi Atzori (PhD, 2000) is professor of Telecommunications at the University of Cagliari where he leads the activities of the MCLab laboratory (Multimedia Communications) with more than 20 research fellows. From 2018, he is the coordinator of the master degree course in Internet Technology Engineering. His research activities focus on the technologies for the realization of Internet of Things (IoT) solutions, with particular reference to the definition of algorithms for the development of social networks among the devices for the realization of Social IoT paradigm. He also works in the area of Quality of Experience (QoE), with the application to the management of services and resources in new generation networks for multimedia communications. Lately, he also applies the study of QoE to IoT services. He is the founding partner of the GreenShare spinoff of which he currently serves as CIO; the company provides IoT services in the sustainable mobility sector. He is a regular reviewer for the EU and for Irish, Spanish and Swedish research programs. He has been the coordinator of European projects on QoE and the IoT (QoE-Net, Demanes and Netergit). He serves regularly in the organizing committees of conferences in the sector and as associate and guest editor in several international journals (IEEE IoT journal, Ad Hoc Networks, IEEE IEEE Open Journal of the Communications Society, IEEE Communications Magazine, etc.).