Intent-based Control Loop for DASH Video Service Assurance using ML-based Edge QoE Estimation

Christian Esteve Rothenberg*, Danny A. Lachos Perez*, Nathan F. Saraiva de Sousa*, Raphael V. Rosa*, Raza Ul Mustafa*, Md Tariqul Islam*, and Pedro Henrique Gomes†
*University of Campinas (Unicamp): {chesteve, dlachosp, nsaraiva, rvrosa, razaul, tariqsaj}@dca.fee.unicamp.br
†Ericsson Research: {pedro.henrique.gomes}@ericsson.com

***Abstract*—Intent-Based Networking (IBN) proposals are based on autonomous closed-loop orchestration architectures that monitor and tune network performance. To this end, IBN defines high-level policies and actions implemented by a closed-loop system. This work demonstrates a Closed Control Loop (CCL) architecture for video service assurance using Machine Learning (ML) based Quality of Experience (QoE) estimation at edge nodes. As part of the solution, network-level Quality of Service (QoS) metrics patterns (*e.g.*, RTT, Throughput) collected through flow-level monitoring are used to build a QoS-to-QoE correlation model tailored to specific target network regions, user groups, and services, in our case DASH video streaming. The demo will showcase the CCL workflow triggering the Orchestrator to take appropriate network-level actions to overcome network QoS degradations and restore the QoE target based on the intent associated with the video service.**

I. Introduction

Novel means for end-to-end network programmability and flexibility through software-defined networking (SDN) and network function virtualization (NFV) require companion approaches of service assurance. However, traditional networks still rely on low-level parameters manually configured by humans as input to specific execution commands [6]. Highly softwarized future networks demand configuration, management, and optimization in an automated manner over the full life-cycle of a service [1].

Driven by demand and technology, the concept of intent-based networking (IBN) is (re)emerging to support efficient and faster service delivery through a high level of automation that simplifies the management with minimum intervention from human operators [8]. In other words, operators only indicate high-level abstracted service requirements (*i.e.*, their "intents"), and the automated system resolves how the intent-based services are executed and validates their compliance.

Service assurance deals with different functions (monitoring, analysis, planning, and execution) that are necessary to ensure that the network complies with the desired intent-based service. More recently, the implementation of such functions has followed the principles of autonomous Closed Control Loops (CCLs) [3]. CCLs establish constant monitoring (*a.k.a.*, data collection / "observe") and SLA verification (*a.k.a.*, analytics) over the entire life-cycle of an intent-based service to assure that it is operating correctly and, if not, to set actions to return to the desired intent. In this context, efforts are being devoted to the applicability of Machine Learning (ML) and Artificial Intelligence (AI) to CCL network and service operations [5]. Scalable and flexible feature monitoring and analytics platforms are required so that adequate ML/AI algorithms can contribute to orchestration functions on the network elements to assure the service in terms of end-to-end quality, fundamentally as perceived by end-users (*i.e.*, QoE).

To contribute towards the realization of CCL architectures, we design an intent-based control loop system platform for service assurance through ML-based QoE estimation from network-level monitoring and/or telemetry information (*i.e.*, QoS metrics). Such metrics are collected and processed at network edge facilities (*e.g.*, as defined by ETSI Multi-access Edge Computing - MEC) strategically located close to the target end-user(s). For validation purposes, this demonstration focus on a DASH video streaming service assurance use case[1]. This demo exercises the loop design principles by implementing all the key components in our proposed architecture (see Fig. 1): (i) *collector* (realizes QoS measurements), (ii) *QoE estimator* (executes ML algorithms), (iii) *policy-driven orchestrator* (enables service life cycle workflows), and (iv) *actuator* (acts on the network).

II. Architectural Overview

In our recent work [7], we proposed an architecture that presents different key characteristics towards providing a flexible and efficient intent-based CCL operation: (i) *modularity*, use of APIs to allow add/remove components, (ii) *adaptive policy*, change the policy conditions in runtime, (iii) *multi-level intent*, define levels of intents in a bottom-up approach, (iv) *topology abstraction*, abstraction of the network topology, and (v) *smart control loop*, management using machine intelligence techniques to obtain more knowledge of network service. Figure 1 shows a high-level view of the CCL architectural design. Details of each component are given below.

Collector: Awareness of network status is indispensable to adjust and control network configurations to satisfy the intent-based service. This component implements a non-invasive network-level measurement method based on traffic mirroring processed by QoS probes running at the mobile edge computing facilities (*e.g.*, MEC). This method has a two-fold

[1]However, our proposed solution (*i.e.*, leverage custom QoE probes at MEC) can be applied to other services beyond video, such as voice, augmented reality, and virtual reality.

978-1-7281-5684-2/20/$31.00

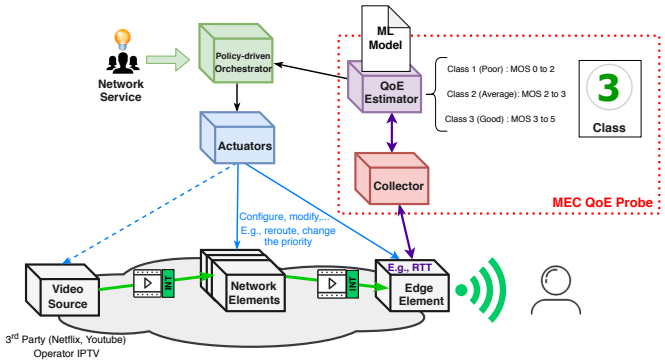


Fig. 1: Architectural Overview.

advantage. First, it does not require endpoints modification or awareness. Second, it does not require on-path middlebox processing. Finally, network-level QoS metrics (*e.g.*, bandwidth, packet loss, delay, etc.) can be tracked in different ways, for instance by using flow-level monitoring/parsing (*e.g.*, PyShark[2], sFlow[3]), or by using telemetry information as P4 INT [4].

QoE Estimator: The applicability of ML methods to output the parameters used by the closed-loop system to continuously adjust the network configuration. The QoE Estimator component performs inferences of Video QoE Mean Opinion Score (MOS) / Key Performance Indicators (KPIs) from network-level QoS metrics received from the Collector component. This QoE estimation can be used for reactive performance diagnosis, run-time network optimization, or even proactive pattern-based network planning. More specifically, this component is based on supervised ML with a multi-variable method used on network QoS metrics. Besides, the QoE estimation is customized to very specific contexts of a target network condition (*e.g.*, congestion in specific backhaul links), topologies, region (*e.g.*, edge, access, core, etc.), a user (*e.g.*, individual user or group), and service (*e.g.*, video streaming).

Policy-driven Orchestrator: This component is based on adaptive policies in a control loop approach that automatically manages a set of orchestrated actions to assure the end-to-end network service quality, fundamentally as perceived by the end-user. The intent-based service is mapped into high-level policies. After that, they are translated into one or more primitive policies that define the metrics to be monitored, thresholds to be evaluated, and control loop operations. The orchestrator abstracts internal configurations (i.e., it is unaware of service details and only knows the service Intent and its status). All the actions and conditions defined by it can be updated at run-time and triggered by any event (*e.g.*, according to the analysis of monitoring network data from the QoE Estimator component). The actions are sent to the actuator component.

[2]https://github.com/KimiNewt/pyshark
[3]https://sflow.org

Actuator: The Actuator component closes the smart control loop by translating the high-level actions sent by Orchestrator to low-level actions, such as network device commands (*e.g.*, API, CLI) of the underlying infrastructure or operations on cloud elements (*e.g.*, VNF, VL). Actuators can perform actions in different scopes, including SDN, NFV, and Legacy networks. Thus SDN controllers (*e.g.*, ONOS, Ryu) or NFV orchestrators can become embodiments of an Actuator.

Functional responsibilities of the Actuator include to inform the Orchestrator, Collector, and QoE Estimator about the performed actions status, *i.e.*, if the action was executed correctly or not. Status information allows the orchestrator to select another action in case of failure.

III. Demo Description

This demo proposal targets DASH video streaming service assurance following a CCL architecture triggered by a QoE inference component based on real-time traffic measurements collected from probes at MEC nodes.

A. Demonstration setup

Figure 2 illustrates the networking scenario under consideration, composed of a video server, which could be located anywhere on the Internet or in the network operator premises, a video client, and the packet flows from/to the video server passing through the Internet, operator network, and access network. Besides, in Figure 2, a candidate MEC server used to run a monitoring agent. Eventually, the video client can also run a monitoring agent.

Our prototype implementation and experimental platform[4] is based on (i) Containernet emulated network infrastructure, (ii) ElasticSearch to store/access network data, (iii) Neo4j graph-based database embodying the annotated topology, (iv) Kibana to generate statistical graphs, (v) Ryu controller as the Actuator component, (vi) Supervised ML model for QoS-QoE correlation in the QoE Estimator component, and (vii) basic Python implementations of the Collector and Policy-driven Orchestrator (CCL workflow engine) components.

B. Demonstration Workflow

The demo is based on a set of pre-demo and live-demo steps, as presented in Figure 3.

Pre-Demo Steps

- *QoE Intent Service Descriptor:* The service descriptor data model (*e.g.*, ETSI NSD/VNFD [2]) is extended to include a target QoE Intent. For the case of video service, three categories are used to quantify the QoE intent: (1) poor, (2) average, and (3) good.
- *ML-based QoE Model building:* In this step, we build an ML model using a Random Forest (RF) classifier with multiple variables for the prediction of QoE metrics. More specifically, we use the MOS values (an objective parametric model) ranging between 0 and 5 as a QoE target according to ITU-T P.1203[5]. We train the model with DASH based

[4]https://github.com/intrig-unicamp/ccl-demo
[5]https://github.com/itu-p1203/itu-p1203

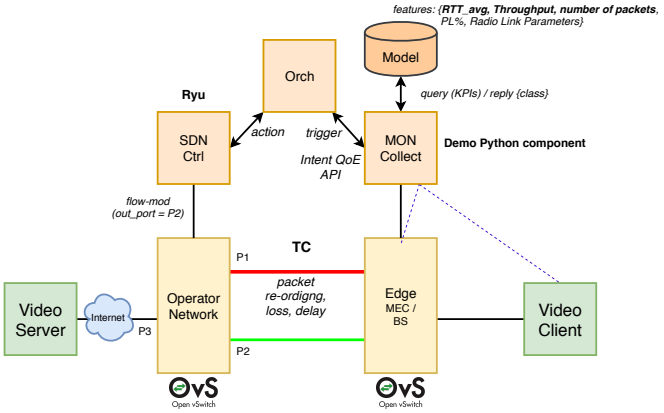


Fig. 2: Demonstration Setup.

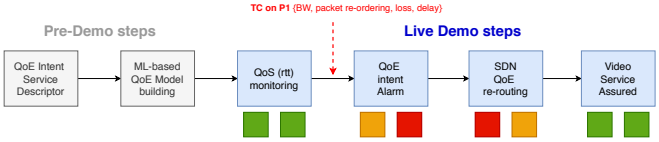
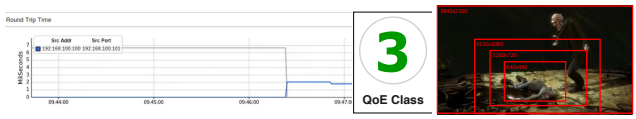


Fig. 3: Demo Storyline



(a) (b) (c)

Fig. 4: Demo Dashboards

per-segment streaming traffic for specific users, network conditions, and network topology. In the end, per video segment based network-level QoS metrics (*e.g.*, uplink RTT, downlink throughput and packets) are used as indicators to estimate with three reasonable precision (*e.g.*, MOS ranging [0, 2] as poor, [2, 3] as average, and [3, 5] as good) end-users' QoE for DASH video service at specific access networks.

Live-Demo Steps

- *QoS Monitoring:* We consider a flow-level monitoring and parsing tool (PyShark) to collect real-time statistics. The tool continuously collects and parses data from the last edge component, *i.e.*, at the network device (Open vSwitch) closest to the video client. Afterward, the ML model uses this data to figure out the video stream QoE.
- *QoE Intent alarm:* The Policy-driven Orchestrator interacts with the Monitoring component to detect per video segment based QoE variations as derived from the QoE estimation model. If the QoE estimated from the QoS metrics (*e.g.*, uplink RTT, downlink throughput, and packets) is not within the QoE intent agreed for that user and service, network control actions are executed (*e.g.*, change the route) through SDN controller. To force video quality degradation (*e.g.*, DASH resolution shifting down, and stalls), we introduce network performance issues by modifying link parameters in path P1 through Linux Traffic Control (TC) scripts. Changes in the link bandwidth, packet re-ordering, packet loss and/or network delay cause variations on the network-level QoS KPI. The CCL system identifies the QoE intent violation and triggers a QoE service restoration request.
- *SDN-based QoE-driven re-routing:* A high-level QoE restoration action is translated to network forwarding intents supported by the SDN Actuator (Ryu controller), which, in turn, performs the state changes to adequately reroute the video streaming to path P2.
- *Video Service Assured:* The video perceived by the end-user returns to the target QoE intent (*i.e.*, QoE category), thus assuring the video service quality.

C. Demo Experience

All the visible dashboards of the demo are shown in Figure 4: (i) a dashboard showing the per video segment live QoS KPIs capture statistics used as input for the ML model (Fig. 4a), (ii) estimated QoE class (poor, average, and good category) from the QoS-QoE correlation ML trained model (Fig. 4b), and (iii) DASH video resolution changes (Fig. 4c).

IV. Conclusion and Ongoing Work

In this demonstration, we present at least four components that are in the current state-of-the-art: Intent at the service level (Video QoE), ML methods at the edge (MEC), DASH-based video streaming (challenging for QoE estimation), and CCL-based automation. Regards the latter, future activities involve the use of P4 programmable switches to gather fine granular telemetry information to build the QoE-QoS model and feed the edge QoE Probe at run-time. Moreover, ML methods will be used not only to correlate but also to predict the overall users' QoE.

References

[1] Nathan F. Saraiva de S., Danny Lachos P., Raphael V. Rosa, Mateus A.S. Santos, and Christian E. Rothenberg. Network service orchestration: A survey. *Computer Communications*, 142-143:69 – 94, 2019.

[2] ETSI Industry Specification Group (ISG) NFV. ETSI GS NFV-SOL 001 V2.5.1: Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification, 2018.

[3] IBM. An architectural blueprint for autonomic computing. Technical report, IBM, 2005.

[4] Changhoon Kim, Anirudh Sivaraman, Naga Katta, Antonin Bas, Advait Dixit, and Lawrence J Wobker. In-band network telemetry via programmable dataplanes. In *ACM SIGCOMM*, 2015.

[5] Tom Nolle. What Role Can AI Play in Service Lifecycle Automation? https://blog.cimicorp.com/?p=3122, 2017.

[6] L. Pang, C. Yang, D. Chen, Y. Song, and M. Guizani. A survey on intent-driven networks. *IEEE Access*, pages 1–1, 2020.

[7] Nathan Saraiva, Nazrul Islam, Danny Alex Lachos Perez, and Christian Esteve Rothenberg. Policy-driven network traffic rerouting through intent-based control loops. In *Proceedings of the XXIV Workshop on Management and Operations of Networks and Services*, pages 15–28, Porto Alegre, RS, Brazil, 2019. SBC.

[8] Qiong Sun, Will LIU, and Kun Xie. An intent-driven management framework. Internet-Draft draft-sun-nmrg-intent-framework-00, IETF Secretariat, July 2019. http://www.ietf.org/internet-drafts/draft-sun-nmrg-intent-framework-00.txt.