Policy-Driven Network Traffic Rerouting Through Intent-Based Control Loops

Nathan F. Saraiva de Sousa[1], **Nazrul Islam**[1], **Danny Alex Lachos Perez**[1], **Christian Esteve Rothenberg**[1]

[1]Department of Computer Engineering and Industrial Automation
School of Electrical and Computer Engineering
University of Campinas (UNICAMP)
Campinas, SP, Brazil

{nsaraiva,nazrulis,dlachosper,chesteve}@dca.fee.unicamp.br

Abstract. *Year after year, the growth of video traffic over the Internet keeps increasing. Video streaming over best-effort networks is considered inefficient and inappropriate to meet the expected Quality of Experience (QoE) of the new generation of multimedia services. Over the past few years, a number of technologies have emerged to improve the state of the art of video delivery, including HTTP Adaptive Streaming (HAS) that adapts the bitrate according to network conditions. At the crossroads, Software Defined Networking (SDN) offers options to meet Quality of Service (QoS) objectives for improved video quality by exploiting end-to-end programmability of network behavior. However, traditional SDN approaches require dealing with low-level details from the underlying infrastructure, interfering in the automation and agility of service deployments. To alleviate these issues and overall provide a simpler approach, Intent-Based Networking (IBN) is being proposed to abstract low-level configurations through high-level policy interfaces. In this paper, we explore such an approach by implementing intent-based control loops for video service assurance. The proposed methods dynamically reconfigure the network for service-specific requirements using IBN to define high-level behavior. We experimentally evaluate a use case where video traffic is rerouted based on network conditions to improve the QoS. The Proof-of-Concept results point to the potential of enhancing video content delivery through QoS-aware Intent-based approaches.*

1. Introduction

User demands for Internet video services keep increasing year after year. According to Cisco's Visual Networking Index, 80% of the world's, mobile data traffic will be video by 2019 [Marshall]. As stated in [Cisco 2017], by 2021, 90% of the global consumer traffic and 78% of mobile traffic will be filled with various forms of video. This growth is the main base for the evolution of new multimedia solutions over the networks. An example is HTTP Adaptive Streaming (HAS), widely used around the globe to provide better video quality. While HTTP streaming presents many advantages to deliver improved QoE, managing over the top video streaming applications possess multiple challenges to network designers and service providers [Singh et al. 2012].

Intent-Based networking (IBN) [Clemm et al. 2018] is an emerging concept in the networking community towards practical ways to interface network management and con-

trol systems through high-level policies untangled from underlying infrastructure specificities. A Network Intent refers to the high level of abstraction where the application logic is expressed in terms *what* should be done through high-level semantics and not *how* it should be done. One example of Northbound Interface supporting Intents is the ONOS controller [Linux Foundation 2018], which uses compilation process for translating the Intents to low-level flow rules.

In addition to use high-level abstractions, network operators seek to automate their management and monitoring processes to reduce the Operational Expenditure (OpEx) and to enhance the network performance. The behavior of Control Loops (CL) can be controlled using a policy-based system which automates the process workflows. Through CL, it is possible to improve service efficiency, providing self-healing and service assurance, reducing OpEx, and increasing revenue through shorter time to market [ETSI 2017]. Relevant open source projects such as ONAP [1] and OSM [2] feature CL mechanisms to manage network services lifecycles.

This paper focuses on applying intent-based control loops to improve QoS of video streaming service. Following policy-driven network management (monitor, analyze, and execute) approach, we automate and abstract the details of the underlying network through and effectively a combination of closed control loop and intent-based networking principles. We present a smart control loop mechanism responsible for managing a set of orchestrated actions on the network elements towards assuring the quality of network services. For validation purposes, we developed a proof of concept prototype leveraging best of breed open source tools capable of rerouting video network traffic based on network conditions, high-level policies, and IBN interfaces. Our approach also introduces an abstraction of the network topology through a graph-based database. Targeting an end-to-end video streaming use case, our experimental evaluation presents the obtained network QoS and video QoE metrics (e.g., PSNR and SSIM).

The remainder paper is organized as follows. Section II describes the background and related work in the areas of SDN, closed control loop and IBN. Section III presents the proposed approach for QoS-aware Intent-based Control Loop. Prototype implementation and experimental platform are shown in details in Section IV. Results and Analysis are discussed in Section V. Finally, Section VI concludes the paper with future work directions.

2. Background and Related Work

2.1. Software-Defined Networking and Closed Control Loops

Software-Defined Networking (SDN) allows operators a flexible and efficient utilization of their infrastructures by a software-centric service paradigm [Kreutz et al. 2015]. However, to realize the standard, operators are required to model the end-to-end service. The operator should have the ability to abstract and automates the control of virtual as well as physical resources to deliver the service. The coordinated set of activities behind such process is commonly referred to as orchestration. Network Service Orchestration (NSO) provides multiple network technology [de Sousa et al. 2018] with a common understanding

[1] https://www.onap.org/

[2] https://osm.etsi.org/

and similar alignment. Traditionally, the leading solutions for video service optimizing have focused on the endpoints by adapting the video, source code (and hence throughput) to the varying network conditions [Wu et al. 2001]. SDN is enabled end-to-end network programmability and allows video service delivery with the new approaches based on active participation of the network.

Consequently, network automation through closed control loops is essential for end-to-end service assurance. Four critical phases are necessary to create a simple closed control loop. Indivisibly, each of these steps takes decisions. The four steps are sequenced, *collect=>analysis=>decide=>execute* [Stein 2018]. This approach is assistance in making decisions to reconfigure the network for service-specific requirement dynamically. The network is adjusted with service and resources as well as a new service offering. Policies are used to manage the services and resources to achieve the desired target [ETSI 2017]. For instance, policies can be controlling the new state of the action and overall action is managed by the closed control loop. In a closed control loop, the Machine Learning (ML) and Artificial Intelligence (AI) techniques are used to control and re-configures the network to improve the maintenance and their applications. Big data and machine learning approach require intelligence to handle a large and varying volume of video traffic [Cui et al. 2016]. In turn, inside of the control loop, an adequate policy will trigger a set of orchestrated actions on endpoints to assure the performance of video service concerning QoS. Thus, how to achieve high video streaming performance is a vital issue in the research and development of NSO solutions.

2.2. Intent-based Networking (IBN)

Intent and Intent-Based Networking are emerging trends [Clemm et al. 2018] closely related to Policy, a well-known concept in networking. The key idea around the IBN is to interface the network by means of an "Intent", i.e., no network implementation is specified, the network itself needs to implement all the necessary actions to fulfill the Intent [Noction 2018]. An Intent specifies the "need" rather than the "how". Intent statements can be descriptive or prescriptive [Kiran et al. 2018]. A descriptive (or declarative) Intent uses high-level statements to specify the problem context, for example, "allow traffic between X and Y". Instead, a more network specific information is considered a prescriptive Intent, for example, "from X:10.0.0.1 to Y:10.0.0.2 set rule=allow".

There are different efforts related to building Intent-based systems solutions. ONOS Intent Framework [Linux Foundation b], ODL Network Intent Composition (NIC) [Linux Foundation 2015], ODL Group Based Policy (GBP) [Linux Foundation a], NEMO [c15] are solution examples using a prescriptive approach where users need some knowledge of the network and Intents are controller-specific which means that they depend on the network controller, constraints need to be converted into technology-specific constraints. In case of descriptive frameworks, architectural designs such as DISMI [Sköldström et al. 2017] and iNDIRA [Kiran et al. 2018] consider a simple, high-level, and technology-agnostic interface, which in turn uses a prescriptive module in order to translate advanced primitives into low-level primitives.

DISMI [Sköldström et al. 2017] provides Intent-based northbound interfaces for network controllers that allow applications to request connectivity (including traffic constraints). The DISMI Intent model defines three different classes of primitives: (i) actions (determine the connectivity type), (ii) nouns (determine what to connect), and (iii)

modifiers (determine how the connectivity should behave). Another framework, iNDIRA (Intelligent Network Deployment Intent Renderer Application) [Kiran et al. 2018] interfaces between the SDN north-bound interface and applications, understanding network QoS requirements, and automating the translation for scientific applications. It uses an ontology-based approach to define queries and translates them to network commands.

Other related work on Intent-based frameworks for specific network applications includes [Sanvito et al. 2018, Comer and Rastegarnia 2018, Han et al. 2016, Tsuzaki and Okabe 2017]. The Open Software Defined Framework (OSDF) [Comer and Rastegarnia 2018] provides high-level APIs that allow managers to express network requirements for applications (network configuration, monitoring, and QoS provisioning) without knowing about low-level details (e.g., network topology, flow rule details). In [Han et al. 2016], the authors propose a software-defined Network Virtualization (NV) that automates the network management and configure the virtual network on the basis of Intent. The advantage of this platform is to provide an automated VN management method with high-level Intent. It also allows management applications with high-level representation. According to paper [Sanvito et al. 2018], the authors propose a process to compile the multiple Intents and re-optimize the paths according to their flow statistics. The Intent is handling to optimize the traffic forwarding of ONOS applications. Further, [Tsuzaki and Okabe 2017] describes an automatic management procedure to update the network configuration based on the Intent with reactive configuration.

2.3. Contributions

In short, this work proposes a system that automatically manages the network service in order to assure and optimize video streaming service. Although our focus of this work is video service, the proposed approach could be used for any services that demand restricted quality metrics. Our work envisions different contributions based on 5 key features: *(i)* modularity, use of API to allow add/remove components, *(ii)* adaptive policy, change the policy conditions in runtime, *(iii)* multilevel 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. Table1 compares our proposed to related works.

Table 1. Related works and associated features

Feature	[Sanvito et al. 2018]	[Comer and Rastegarnia 2018]	[Han et al. 2016]	Our proposal
Modularity	YES	NO	YES	YES
Adaptive Policy	YES	YES	NO	YES*
Multilevel Intent	NO	YES	YES	YES
Topology Abstraction	NO	NO	YES	YES
Smart Control Loop	NO	NO	NO	YES*

* *Future work*

3. Proposed Approach: QoS-aware Intent-based Control Loops

Network programmability through paradigms such as SDN and NFV allows for new approaches based on active participation of the network to assure the network service delivery. In support of the decision making to reconfigure the network for service-specific requirements dynamically, a large amount of various input data can be exploited. Due to the dynamicity and large volume of such data, big data and machine-learning approaches

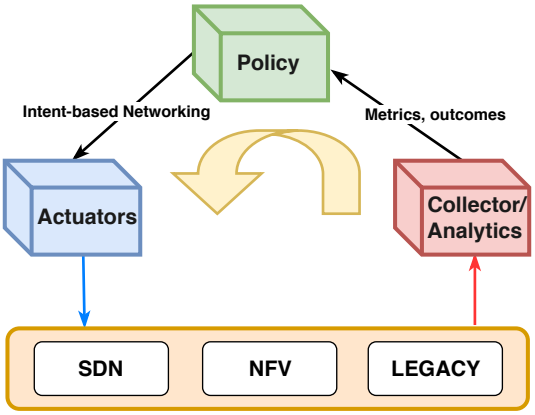


Figure 1. High-level representation of a closed control loop design.

appear as promising approaches for delivering the required intelligence [Cui et al. 2016]. To that end, smart control loop approaches promise to automatically manage a set of orchestrated actions on the network elements (and endpoints if possible) to assure the network service regarding end-to-end quality, fundamentally as perceived by the end-user.

Thus, our objective is to develop a suitable actuation mechanism that implements a closed control loop and translates the user needs and business targets into technology-specific configurations. The system is responsible for assuring the requirements to deliver high-quality video streaming service. The control loop enables the self-healing in end-to-end video services, without human intervention, providing automation and agility in the service management. Figure 1 presents a high-level view of the closed control loop design to monitor and keep the services working based on pre-established agreements, as well as optimize them. In order to abstract the underlying infrastructure through high-level actions and to automate the control loop, we follow principles from IBN.

IBN scopes a higher level of abstraction. For this purpose, we define two layers of intents in a bottom-up approach. In first level, intent assurances network connectivity, e.g., connect two points through different networks. In second level, intents trigger policy-driven actions based on holistic view of service. The Intents enables actions not only on network but also on VNFs and endpoints, including scaling of VNFs, functions placement and changing bitrate.

Briefly, our approach generates a flow of data between the various components that allows implementing an automation system and self-healing of end-to-end network services. In other words, the system detects an issue on a service, and it automatically defines actions to fix it according to pre-set parameters (e.g., SLA). In the next subsections, we explain each component and their functions.

The closed-loop system introduces a delay in monitoring and decision making. However, the system will work with two lines of action: corrective actions, in case of serious failures or SLAs violation, and preventive actions, detected from the Analytics component and avoiding possible future issues.

3.1. Collector/Analytics

The Collector component monitors the deployed service (e.g., video stream) collecting QoS and QoE metrics from SDN and no-SDN data plane. Firstly, it is required to define a set of metrics (e.g., bandwidth, packet loss, CPU) among all those available. Such metrics will enable to register the service performance. After, the data are organized in a well-defined information model for providing service-related information. Differently from works as [da Costa Filho et al. 2018] and [Comer and Rastegarnia 2018], the monitoring of the services is done through the installed intents for each network flow resulting in more precise monitoring.

The collected data are stored in a database to track the historical data and on-line and off-line analysis in runtime. The database stores the status of infrastructure (e.g., topology, components), deployed services and relationship among them. Besides, it must have high performance, scalability, and availability because of the dynamics and the large volume of network data. The same element can also run data analytics based on machine learning (ML) and artificial intelligence (AI) techniques. It analyzes the collected data and exposes parameters and processed information to Policy component. Also, it receives information from performed actions, from there it generates reports and feedback to *Policy* to facilitate the service/infrastructure adjustment. Through historical network data, it is possible to create a tendency curve and an original curve. When a threshold is exceeded in the tendency curve, an alert is sent to the Policy component.

3.2. Policy

Formally, policies are defined as a set of rules that are used to manage and control the state of one or more managed objects. In the control loop context, policies can manage the interaction among services and resources with the environment to obtain the desired objective [ETSI 2017]. There are three types of policies: *(i) Imperative* – uses statements to define the state of targeted objects explicitly. Thus, a set of tasks are executed in the correct order using an event-condition-action tuple; *(ii) Declarative* – expresses the targets of the policy, but not how to achieve it. It follows a formal logic without define clearly the objects state. Examples are SQL and OpenStack Congress[3]; and *(iii) Intent* – expresses the goals of the policy in high-level. It has not formal logic. Thus it needs the translation to one or more primitive policies.

The Policy component is responsible for defining and applying policies that ensure the quality of service. For that, it uses the metrics and information provided by both the Analytics component and database. It abstracts internal configurations as well, i.e., it is unaware of service details, only knows the service Intent and its status. Its operation consists of the monitoring of different metrics and events (e.g., end-to-end data plane video sessions) and reacts by the execution of programmable workflows. The workflow consists of a set of conditions that can be updated in runtime and is triggered by any events (e.g., when the flow has packet loss is greater than 10%). The component always actuates according to networking goals.

We propose a policy that outlines the logic involved in the process of network service management as depicted in Figure 2. It is an extended version from imperative

[3]https://docs.openstack.org/congress/

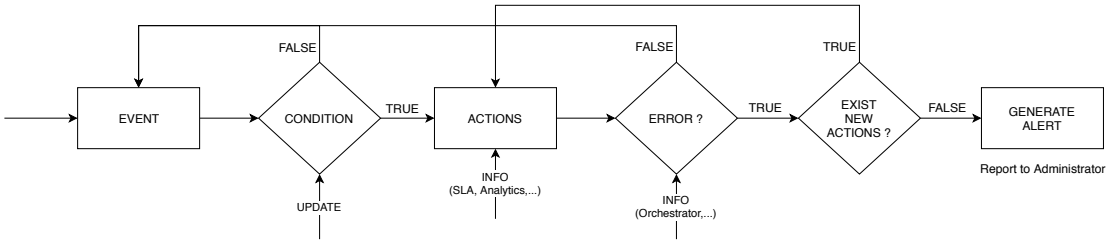


Figure 2. Workflow representation of the proposed extended imperative policy.

policy and describes a simple workflow to implement self-healing in case of error in the execution of the actions. Standard Event-Condition-Action (ECA) policies do not have a detection and recovery mechanism in case of failure to perform actions as we propose.

The workflow starts when an event (e.g., metric threshold) is identified by Policy component. This event is compared to a condition (e.g., traffic is consuming 80% of the link), if it is false, so the system returns to listening state; otherwise, the system lists a set of actions to be performed. The condition can be updated in runtime. After that, the system defines one or more actions to be executed based on information from external elements such as Analytics. If there are no errors in the execution of the operations, the flow returns to the listening state. Otherwise, another action is chosen from the list. This process loops until the system find an action that is executed correctly, or there are no more possible actions. In this case, the system generates an alert reporting to the network administrator.

3.3. Actuators

The Actuator component closes the smart control loop by translating the high-level policies sent by Policy to low-level actions, for instance, OpenFlow flows. The policies are expressed in a specific language closer to the natural based on an Information Model, and require a mapping to network device commands (e.g., API, CLI) of the underlying infrastructure.

Actuators can perform actions in different scopes including SDN, NFV, and Legacy networks. Thus SDN controllers (e.g., OpenDaylight, ONOS), NFV orchestrators or Management Systems can become embodiments of an Actuator. The other components of the architecture are decoupled from the actuators, which are integrated through the northbound interface (NBI). Functional responsibilities of the Actuator include to inform the Policy about the performed actions status, i.e., if the action was executed correctly or not. Given status will enable the Policy to select another action in case of failure and seed the Analytics component towards improving the outcomes.

4. Prototype Implementation and Experimental Platform

Figure 3 illustrates our prototype implementation and experimental platform based on (*i*) Mininet 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*) ONOS[4] controller, and (*vi*) proposed IBN/CL Architectures

[4]https://onosproject.org/

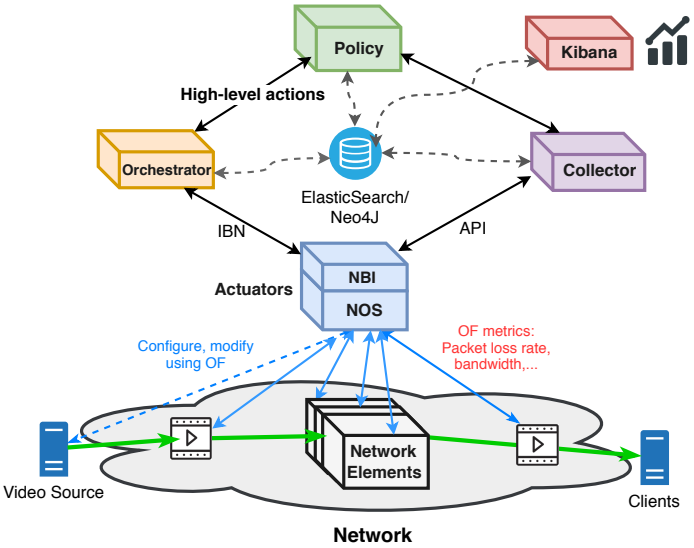


Figure 3. Prototype Implementation and Experimental Platform

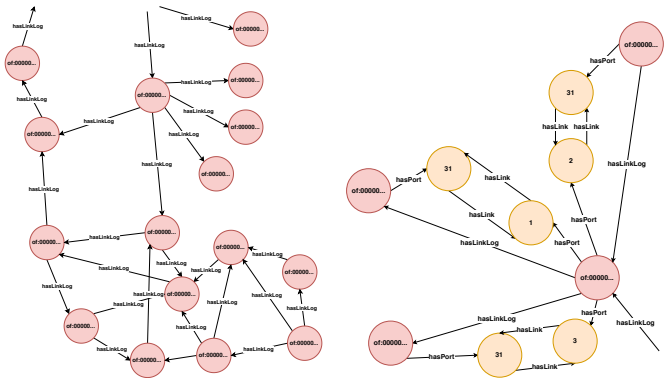
based on the Policy, Collector, and Orchestrator components. Its objective is to reroute network traffic based on network conditions, high-level policies, and IBN interfaces in different scopes (high-and low-level intent). Note that the component Actuator from reference architecture was split into ONOS and Orchestrator. ONOS APIs are used to collect information on the network such as topology, nodes, links, and intent status.

The Collector is a Python-based component responsible for collecting and storing the data from the network. The design requires just one-way measurements of links used by network services. Currently, it collects bandwidth, packet loss, and hop count metrics through Intent Monitor and Reroute (IMR) application from ONOS API. The IMR exposes statistics and re-routing capabilities, of specific Intents, to external applications. There is also a java application that captures topology information and stores them into a graph-oriented database, Neo4J[5]. The topology is stored into Neo4j graph model as shown in Figure 4. In this prototype version, the Analytics component is not implemented.

All network measurements are realized per flow and per link (between switches) and stored into ElasticSearch[6]. Accessible through an extensive and elaborate API, Elasticsearch allows for quick searches in support of data discovery applications. Kibana[7] easies the visualization of Elasticsearch data by means of rich graphics. ElasticSearch and Kibana are highly scalable, well-documented and easy to set up experimental scenario, and it is being used in important projects including ONAP and OSM.

The Policy component is implemented using Python language too. It consumes data directly from ElasticSearch or Collector and realizes a verification of Key Performance Indicators (KPIs). This prototype uses a simple imperative policy, no-extended yet, with three-tuples, consisting of an event, a condition, and an action clause. Pol-

[5]https://neo4j.com/
[6]https://www.elastic.co/elasticsearch
[7]https://www.elastic.co/kibana



(a) Switches (b) Links (Switches + Ports)

Figure 4. Abstracted topology into Neo4J, e.g., switches and links.

icy checks a set of metrics periodically and compares them with some pre-set condition (e.g., packet loss is higher than 5%). If the condition is satisfied, then the Policy sends high-level action to Orchestrator. Examples of high-level action are: *select shortest path avoiding link "X-Y"*, or "*Route traversing links with properties (A, B, C)*."

The Orchestrator, in turn, is a Python-based application that works as a Broker that handles technological details related to network service management. Its function is to translate high-level actions sent by Policy to primitive Intents of ONOS, which are further decomposed into simpler actions (e.g., install OpenFlow flows). It can work as a plugin for the orchestration platform as OSM. Therefore, all actuation is implemented via northbound Intent interface from ONOS. Components consume database information updated at runtime.

5. Experimental Validation and Evaluation

For validation and evaluation through experimentation, we developed a Proof-of-Concept use case scenario based on a video streaming service. We first analyze the connectivity between hosts using `ping`. We then start the video stream from a video server to one or more clients. The Control Loop system monitors the video service and automatically actuates if a problem is found regarding the intended QoS. To this end, the Collector component periodically monitors the bandwidth, packet loss rate, and hop count per flow and per link.

Figure 6 presents the evaluation scenario: network topology composed of 34 devices, organized following an operator-like access-aggregation-core topology, inspired by [da Costa Filho et al. 2018], featuring up to 20 clients connected to the edge switches and one video server. The idea of this scenario is to represent a video server located in the Internet streaming video to client 1 in the access network. VLC [8] is used as the video software.

For the sake of our PoC scenario, the policy is applied over a unique metric: packet loss rate. Considering "Intent" as the desired state of an action to automate the network control, the "Intent" is applied whenever packet loss of monitored flow is greater than X value, e.g., 12%. After identifying the problematic network segment, the system selects

[8] https://www.videolan.org/vlc/index.html

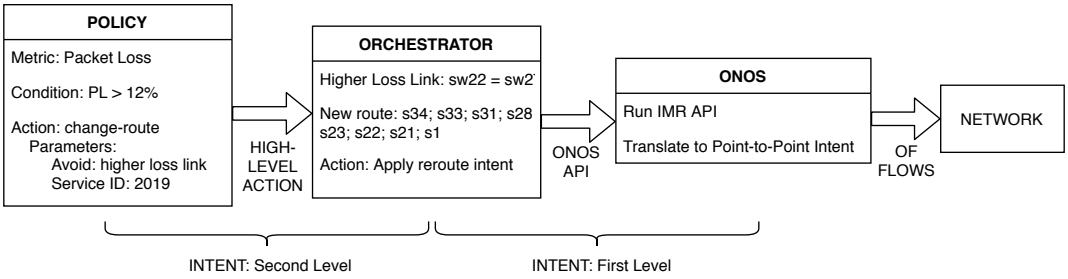


Figure 5. Translation of high-level policies to underlying configurations.

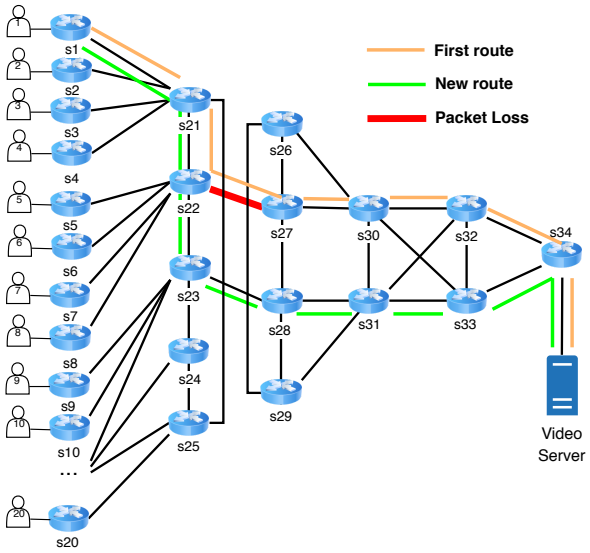


Figure 6. Topology used in the experimental evaluation. Adapted from [da Costa Filho et al. 2018]

the next shortest path avoiding such link. Annotated topology state is kept in the Neo4J database to provide proper routing data. Figure 5 presents detail on the process of policy enforcement until the effective action execution. The Policy sends a high-level action to the Orchestrator to effectively reroute the traffic. The Orchestrator translates the high-level action to Intents supported by the ONOS controller, which, in turn, performs the state changes to adequately reroute the traffic between the video server and the client.

The initial route between the video server and the client (1) goes through on $sw34- > sw32- > sw30- > sw27- > sw22- > sw21- > sw1$. The Control Loop monitors throughput and packet loss metrics. The experiment script uses TC [9] to configure the packet loss to force a 3% loss in the link between $sw22$ and $sw27$. The system identifies the packet loss, but the Policy performs no action since the rate is lower than 12%. Next, the packet loss rate is increased to 13%. In this case, the Policy notes that the threshold was exceeded. After querying ElasticSearch to identify the under-performing link ($sw22 = sw27$), the Orchestrator is requested to reroute the flow to avoid the said link by querying Neo4j to return all shortest routes except that link. The new route is selected and through ONOS APIs the route changed by the IMR and IFWD applications. The new path looks as follows: $sw34- > sw33- > sw31- > sw28- > sw23- > sw22- >$

[9] http://man7.org/linux/man-pages/man8/tc.8.html

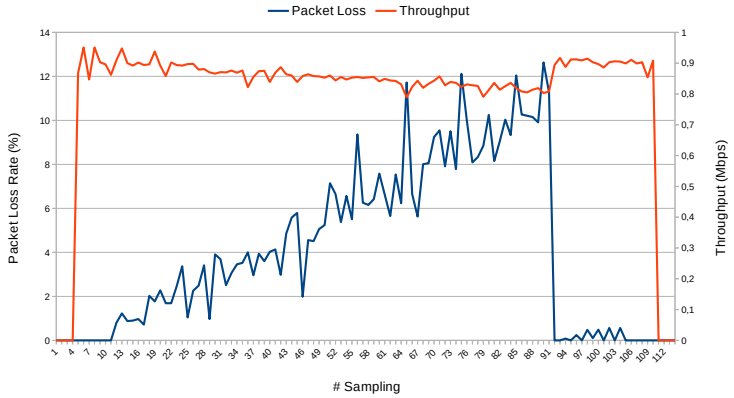


Figure 7. Packet loss and throughput before and after applying the policy.

$sw21- > sw1$. Note that the ONOS Intent framework provides fault-tolerance and resilience even when some switch fails, i.e., the Intent is recompiled by rerouting the flow in the case of failure. If there is no alternative path, the framework tries to establish the path once one becomes available.

5.1. Results and Analysis

Evaluation of QoS metrics In this experiment, the video duration is 3767 seconds and the packet loss in link $sw22 = sw27$ is increased by 1% every 280 seconds. We set up the policy to applying a rerouting action when the flow packet loss is greater than 12% and the polling interval is 35 seconds. Figure 7 presents the observed packet loss and throughput during the execution of the experiment.

Note that the packet loss starts in 0% and gradually increases until it exceeds the limit, around sample #93. Then, the flow is rerouted to another path with no loss, and from that point, the loss drops to practically 0%. Observe that the throughput decreases as losses increases, returning to normal conditions after the policy is applied. This behavior is normal due to the number of packets dropped.

Evaluation of QoE metrics To evaluate the quality of the video, we use the traditional PSNR and SSIM metrics. PSNR can be interpreted as a measure of the peak error whereas SSIM provides a metric of the overall video quality. Video transmissions with higher PSNR and SSIM closer to one (1) indicate better quality of the video. To provide the PSNR and SSIM QoE metrics, we use the Evalvid.[10] The experiment workflow is similar to the previous one. In this case, another video, composed of 15691 frames, is used and the packet loss is manually configured. Figure 8 presents the experiment results, where the X-axis represents the frame number, and the Y-axes represent the PSNR and SSIM.

We can observe no packet loss between frames 1 to 1460 with PSNR ranging between approximately 35 and 50, and SSIM close to 1. The packet loss between 1 to 10 % corresponds to frames 1460 and 7460, where both PSNR and SSIM float and yield low values. After frame number 7460, the policy is applied, and the path is changed to another one without losses, following the noteworthy enhancements in the QoE metrics.

[10]http://www2.tkn.tu-berlin.de/research/evalvid/fw.html

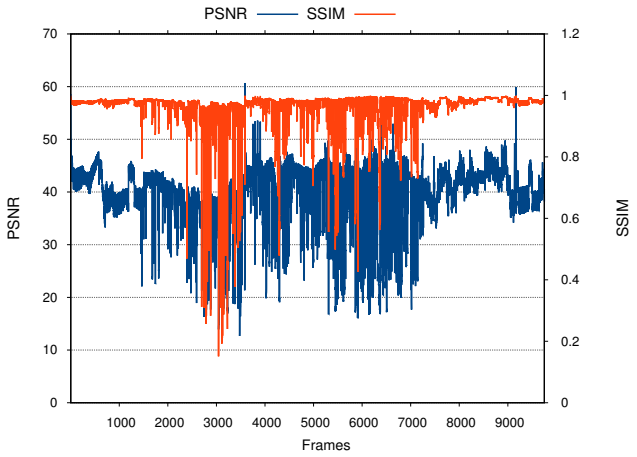
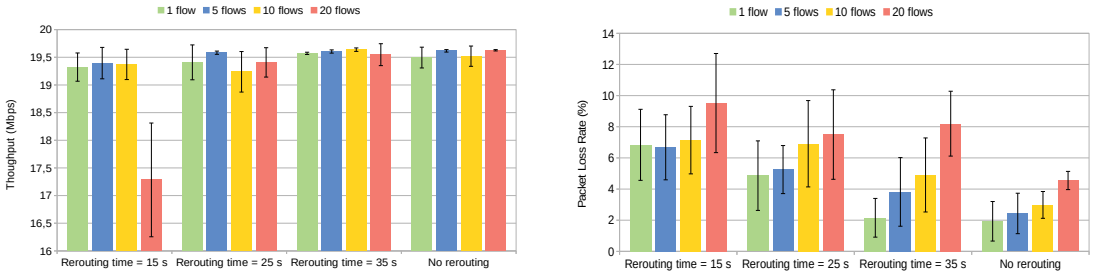


Figure 8. Evaluating QoE metrics: PSNR and SSIM.



(a) (b)

Figure 9. Evaluation of rerouting in terms of (a) Throughput, and (b) Packet loss rate for different flows and rerouting time.

Impact of Control Loop on Network-level Metrics We now analyze the network impact of the closed control loop in case of rerouting events. The goal is to investigate how the network reacts, in terms of packet loss and throughput, when occur rerouting. To this end, we use the same topology of Fig. 6, run *iperf* to generate traffic between *client 1* and the *video Server* and compare traffic in the presence of rerouting to the same traffic without rerouting. A script is used to trigger the route changes between the orange path and the green path. We evaluate the observed behavior for different parallel flows (5, 10 and 20) and different rerouting intervals (15 s, 25 s, and 35 s) over a 350 s simulation. The results are shown in Fig. 9 with 95% of Confidence Interval and ten repetitions.

Regarding throughput (see Fig. 9(a)), the results are statistically equivalent due to the overlapping confidence intervals in all scenarios, except for the case there of 20 simultaneous flows and rerouting interval of 15 seconds. In this case, the frequent changes between the primary and backup paths drop the throughput significantly and with large fluctuation (i.e., high confidence interval). Figure 9(b), presents the packet loss results to assess the impact on performance. The observed behavior is similar to all cases, the larger the amount of parallel flows and lower the rerouting interval, the higher the losses. Various factors influence the performance hit. As one would expect, the crucial one is the

routing changes. The system requires time to apply all new Intents, thereby introducing delay and packet losses in the flows, most critically for large amounts of flows.

6. Conclusions and Future Work

The driving theme of this paper is the potential role of smart control loops and Intent-based Networking concerning automation, abstraction, and self-healing in end-to-end service. We developed a smart control loop approach to improve video service quality. A policy system is exercised to reconfigure the network using IBN as dynamic high-level decisions. The results demonstrated that when the packet loss crosses certain threshold levels, traffic is effectively rerouted thereby enhancing the observed QoS (e.g., Throughput) and QoE metrics (e.g., PSNR, SSIM).

In future work, we will investigate different network topologies and video server locations, including wireless clients using Mininet-WiFi. Parallel work is ongoing on to apply machine learning techniques to identify and classify the traffic to be monitored and improved the smart control loop decisions beyond static packet loss to QoE-centric metrics, including forecast predictions.

Acknowledgment

This research is supported by the Innovation Center, Ericsson S.A., Brazil, grant UNI.64.

References

Nemo - The Application's interface to Intent Based Networks. http://www.nemo-project.net/.

Cisco (2017). Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper - Cisco. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html.

Clemm, A., Ciavaglia, L., and Granville, L. (2018). Clarifying the Concepts of Intent and Policy. Internet-Draft draft-clemm-nmrg-dist-intent-01, IETF Secretariat. `http://www.ietf.org/internet-drafts/draft-clemm-nmrg-dist-intent-01.txt`.

Comer, D. and Rastegarnia, A. (2018). OSDF: A Framework for Software-Defined Network Programming. *2018 15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 1–4.

Cui, L., Yu, F. R., and Yan, Q. (2016). When Big Data Meets Software-Defined Networking: SDN for Big Data and Big Data for SDN. *IEEE Network*, 30(1):58–65.

da Costa Filho, R. I. T., Lautenschläger, W., Kagami, N., Luizelli, M. C., Roesler, V., and Gaspary, L. P. (2018). Scalable QoE-aware Path Selection in SDN-Based Mobile Networks. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, pages 989–997.

de Sousa, N. F. S., Perez, D. A. L., Rosa, R. V., Santos, M. A. S., and Rothenberg, C. E. (2018). Network Service Orchestration: A Survey.

ETSI (2017). Improved operator experience through Experiential Networked Intelligence (ENI). Technical Report 22, ETSI.

Han, Y., Li, J., Hoang, D., Yoo, J., and Hong, J. W. (2016). An Intent-Based Network Virtualization Platform for SDN. In *2016 12th International Conference on Network and Service Management (CNSM)*, pages 353–358.

Kiran, M., Pouyoul, E., Mercian, A., Tierney, B., Guok, C., and Monga, I. (2018). Enabling Intent to Configure Scientific Networks for High Performance Demands. *Future Generation Comp. Syst.*, 79:205–214.

Kreutz, D., Ramos, F. M. V., Veríssimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-Defined Networking: A Comprehensive Survey. *Proceedings of the IEEE*, 103(1):14–76.

Linux Foundation. Group Based Policy User Guide. https://docs.opendaylight.org/en/stable-fluorine/user-guide/group-based-policy-user-guide.html.

Linux Foundation. ONOS Intent Framework. https://wiki.onosproject.org/display/ONOS/ Intent+Framework.

Linux Foundation (2015). ODL Network Intent Composition. https://wiki.opendaylight.org/view/Network_Intent_Composition:Main.

Linux Foundation (2018). ONOS - A new carrier-grade SDN network operating system designed for high availability, performance, scale-out. https://onosproject.org/.

Marshall, C. By 2019, 80% of the World's Internet Traffic Will Be Video. https://tubularinsights.com/2019-internet-video-traffic/.

Noction (2018). Intent-Based Networking Explained. https://www.noction.com/blog/Intent-based-networking-ibn-explained.

Sanvito, D., Moro, D., Gulli, M., Filippini, I., Capone, A., and Campanella, A. (2018). ONOS Intent Monitor and Reroute Service: Enabling Plug and Play Routing Logic. *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, pages 272–276.

Singh, K. D., Hadjadj-Aoul, Y., and Rubino, G. (2012). Quality of Experience Estimation for Adaptive HTTP/TCP Video Streaming using H.264/AVC. In *2012 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 127–131.

Sköldström, P., Junique, S., Ghafoor, A., Marsico, A., and Siracusa, D. (2017). Dismi - an intent interface for application-centric transport network services. In *2017 19th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4.

Stein, Y. (2018). Closed-loop Implementation – one size does not fit all - TM Forum Inform. https://inform.tmforum.org/digital-transformation-and-maturity/2018/03/closed-loop-implementation-one-size-not-fit/.

Tsuzaki, Y. and Okabe, Y. (2017). Reactive Configuration Updating for Intent-Based Networking. In *2017 International Conference on Information Networking (ICOIN)*, pages 97–102.

Wu, D., Hou, Y. T., Zhu, W., Zhang, Y.-Q., and Peha, J. M. (2001). Streaming video over the internet: approaches and directions. *IEEE Transactions on Circuits and Systems for Video Technology*, 11(3):282–300.