Resilience of Delay-sensitive Services with Transport-layer Monitoring in SD-WAN

Sebastian Troia, Marco Mazzara, Marco Savi, Ligia Maria Moreira Zorello, and Guido Maier

***Abstract*—Today, more and more enterprises are embarking on a digital transformation where most of their applications are hosted in the Cloud. As a result, a reliable Wide Area Network (WAN) has become a primary need to interconnect their distributed branch offices and data centers that accommodate those applications. Software-Defined Wide Area Network (SD-WAN) represents the most promising technology solution for next-generation enterprise networks, being able to increase network agility and reduce costs. In this paper, we present an experimental SD-WAN solution capable of running and optimizing delay-sensitive high-priority services, such as real-time video streaming, while minimizing downtime caused by network failures. This solution comprises a monitoring and a traffic engineering system for SD-WAN. The first consists of a Transport-layer Passive Monitoring (TPM) system based on extended Berkeley Packet Filter (eBPF) technology with the goal of monitoring TCP flows; the second consists of an application, running inside the SD-WAN controller, with the goal of orchestrating the network traffic in consideration of the monitoring measurements by ensuring rapid recovery and resilience in case of unexpected congestion events. We validate our solution over two SD-WAN testbeds: the first is hosted in our laboratory at Politecnico di Milano, while the second is deployed in a municipal network of an Italian city. Results show that our SD-WAN solution can increase the overall service availability while meeting the stringent QoS requirements of delay-sensitive services.**

***Index Terms*—Software Defined Wide Area Network (SD-WAN), Software Defined Networking (SDN), Edge Networking, Datacenter Networking, Traffic Engineering, TCP, Network Monitoring, eBPF.**

I. Introduction

Digital transformation is driving organizations to reinvent the way they do business. They are launching products and services to customers faster, thanks to Cloud-based technologies, such as web hosting services. However, traditional enterprise networks (ENs) are not suited to a Cloud-centric world and are struggling to keep up with this change. Many of today's wide area network (WAN) architectures rely primarily on a traditional hub-and-spoke architecture that connects sites to a limited number of regional or private data centers, making it difficult to manage Cloud migration or cope with broadband applications, such as videoconferencing or real-time video streaming.

Sebastian Troia and Guido Maier are with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano and SWAN networks, Milan, Italy. Marco Mazzara and Ligia Maria Moreira Zorello are with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB), Politecnico di Milano, Milan, Italy. Marco Savi is with the Department of Informatics, Systems and Communication, University of Milano-Bicocca, Milan, Italy. Corresponding author e-mail: sebastian.troia@polimi.it. A preliminary version of this paper appeared in [1], presented at IEEE MeditCom in 2021.

Decades ago, the geographical distance between sites of enterprises was bridged by using dedicated lines leased from network operators. Usually, such leased lines had high costs and could guarantee a limited network bandwidth [2]. As a consequence, many different technologies have been proposed to create the inter-site connections of ENs as an overlay over public WANs, such as Asynchronous Transfer Mode (ATM), Frame Relay (FR) and Multi-Protocol Label Switching (MPLS). In particular, the most recent MPLS is currently commonly adopted for its capability of guaranteeing Quality of Service (QoS) according to Service Level Agreements (SLAs) by setting up Label-Switched Paths (LSPs) through the IP network. On the other hand, the operational complexity of MPLS is directly related to the number of branch offices. Although MPLS can be regarded as a milestone in ENs, its high cost and complexity have recently pushed companies to seek hybrid WAN solutions, which simply add a broadband Internet connection or virtual private networks (VPNs) to the current EN architecture [3]. This solution can improve the user experience and Cloud access, but it is cumbersome to configure and totally unreliable to meet changing needs.

Software-Defined Wide Area Network (SD-WAN) can provide the agility and flexibility needed to support the aforementioned digital transformation. SD-WAN offers numerous advantages in terms of high network agility, cost savings, high availability, easier and safer management of the EN. Based on a software-defined architecture, SD-WAN delegates the control and management to a centralized controller connected uniquely to the edge devices, or Customer Premises Equipment (CPE): this means that there is no need to have direct access to the WAN internal devices (e.g. providers routers and switches) to operate an SD-WAN; as such, VPNs over best-effort Internet connections can be used to ensure a certain quality level to applications' traffic.

Moreover, one of the primary benefits of an SD-WAN is its ability to guarantee application-aware traffic routing, that is, it can dynamically allocate network resources and capacity to meet different IT services and applications. For instance, delay-sensitive applications (such as real-time video streaming) require that a certain QoS, in accordance to specific SLAs, is guaranteed to run correctly. SD-WAN allows to monitor their performance and dynamically route the applications' traffic to meet such SLAs. Indeed, if an application's packet routing over an Internet-based VPN cannot guarantee meeting its SLA, the SD-WAN has also the capability to reroute packets over a guaranteed Internet connection such as the one provided by MPLS. Or, if a low-priority application is consuming too much bandwidth over the MPLS connection, the SD-WAN can

reroute packets via an Internet-based VPN to free capacity for higher-priority traffic. The goal is to ensure that each application has the most appropriate quality path to provide optimal customer experience. In other words, enterprises can orchestrate their traffic in consideration of the monitoring measurements on network and service performance, such as packet delay, loss, jitter, and service requirements. Additionally, SD-WAN supports a new way of managing the application traffic thanks to the possibility of instructing the CPEs on the basis of heterogeneous information such as the position of the CPEs, the type of services, characteristics of the flows (TCP, UDP), etc.

However, one question arises: *to what extent is SD-WAN able to ensure guaranteed service availability by exploiting broadband Internet?* This is the problem we investigate with our work. Given the small number of independent studies we found in literature on such a topic, our opinion is that the question is still largely open. In order to fill this gap, in this work we developed an open-source SD-WAN solution based on an innovative monitoring system with the aim of guaranteeing fast recovery and network resilience in the case of network failures.

Specifically, in this paper we focus on optimizing delay-sensitive services, such as real-time video streaming and online gaming. These services continue to be in huge demand in today's homes, as such, by 2022 video streaming will account for 82% of all Internet traffic in the world [4]. Nowadays, most of the video streaming applications are based on HTTP/2 [5], which has been widely used for providing uninterrupted video streaming services over harsh network conditions and heterogeneous devices [6]. In addition, it can easily traverse firewalls and reuse the already deployed HTTP infrastructure such as HTTP servers, HTTP proxies, and Content Delivery Network (CDN) nodes [7]. Given the fact that HTTP/2 relies on TCP, the idea is to build an SD-WAN solution able to monitor and collect real-time transport network information acquired directly from the servers (or hosts) that generate the network traffic. To do so, we exploited extended Berkeley Packet Filter (eBPF) [8], a novel technology that can run programs into the Linux kernel without changing kernel source code or loading kernel modules. Thanks to this technology, we are able to code specific programs from the user space to monitor different parameters of transport network protocols, such as the number of TCP retransmissions of traffic flows. This allows us to access network traffic information from the user space quickly and in real time.

In our previous work in [1], we presented a demo-test running an experimental SD-WAN solution, by means of eBPF, capable of recovering a video streaming service affected by a network failure. Thanks to a first implementation of eBPF, we could quickly identify TCP retransmissions and recover the video flow in real-time.

In this work, we develop an application inside an SD-WAN controller based on ONOS [9], called SD-WAN Traffic Engineering (TE), which interacts with our monitoring system to ensure the best availability to delay-sensitive services and guarantee network resilience. TE is crucial for network availability and reliability. Enterprises can use TE algorithms to orchestrate their traffic among the CPEs in consideration of the monitoring measurements of WAN performance, such as packet delay, loss, jitter, and service requirements. We performed experimental measurements through two real testbeds to validate our solution. The first one is hosted in our laboratory at Politecnico di Milano, while the second has been developed in a municipal network of an Italian city. Specifically, in the latter case we adopted our solution to connect the city hall to a remote office. Experimental results demonstrate the capability of our proposal to increase the overall SD-WAN performance by providing fast recovery in case of congestion events and link failures.

With respect to our previous work in [1], we improve our solution and make the following contributions:

- Design and implementation of a Transport-layer Passive Monitoring (TPM) system based on eBPF with the goal of monitoring all TCP flows;
- Development of a traffic engineering application, called SD-WAN TE, running inside the ONOS controller. This application interacts with the proposed TPM system to speed up the service recovery from WAN failures;
- Deployment of our SD-WAN solution made by the SD-WAN TE and the TPM system over a lab testbed and a real Municipal network of an Italian city called Militello in Val di Catania.

The remainder of this paper is organized as follows:

- Section II presents the related work;
- Section III presents the SD-WAN solution made by SD-WAN TE and TPM system;
- Section IV provides a description of the testbeds used to validate our solution;
- Section V shows the experiments and the results;
- Section VI provides a discussion on the limitations of this paper and some future work;
- Section VII concludes the paper.

II. Related work

Commercial SD-WAN solutions [20] use different traffic engineering techniques to manage the network traffic among the CPEs. Most of them rely on active or passive monitoring (or both) proprietary systems to ensure reliability and resilience. However, implementation details of these solutions are not disclosed or open-sourced. The increased attention for SD-WAN across the enterprise landscape is pushing the academic research world to investigate new solutions for SD-WANs architecture and its management/optimization features. Below, we survey the current research works on two aspects: 1) SD-WAN solutions and their characteristics; 2) TCP-enabled passive monitoring techniques.

A. SD-WAN solutions and their features

In [21] and [22], the Google's network infrastructure team shows how SDN can be exploited to optimize their Google's internal WAN. This WAN is fully controlled by an SDN controller and connects a dozen of data-centers across the planet. It has some unique characteristics: massive bandwidth

TABLE I: Review on the related work about traffic engineering and monitoring in SD-WAN

Ref.	Objectives	SD-WAN applications	Services	Implementations	CPEs - tunnels	Metrics
[10]	Techno-economic analysis	-	ATMs	-	2 CPEs - 2 tunnels	-
[11]	Minimize traffic disruptions and reconfiguration costs	Traffic engineering Active monitoring	Generic	Simulation: 2 network topologies	1) 6 CPEs - 1 WAN (14 nodes) 2) 64 CPEs - 1WAN (48 nodes)	Num. disrupted flows Reconfiguration costs
[12]	TE based on traffic classification and prioritization	Traffic engineering	Generic	Emulation: Openflow-based switch HP 5900	3 CPEs - 5 tunnels	Goodput
[13]	TE based on prioritization to guarantee VoIP traffic under QoS constraints	Traffic engineering	VoIP	Emulation: Mininet software	2 CPEs - 2 tunnels	Bandwidth, latency, jitter, CPU load
[14]	Minimize inter-domain traffic transit expenses	Traffic engineering	Generic	Emulation: 1) Openvswitch and VyOS 2) Juniper MX-240	2 CPEs - 4 tunnels	Throughput
[15]	Demonstration of 1:1 protection scheme for SD-WAN	Traffic engineering	Generic	Emulation: Mininet software	2 CPEs - 2 tunnels	-
[16]	Implementation of MPTCP over SD-WAN	Traffic engineering	Generic	Emulation: 1) lab appliances (servers and local netws.) Real-world: 2) GENI and Amazon AWS	1) 3 CPEs - 5 tunnels 2) 3 CPEs - 2 tunnels	Throughput, latency
[17]	TE based on critic-only reinforcement learning and active tomography to reduce end-to-end traffic delay	Traffic engineering Active monitoring	Generic	Simulation	2 CPEs - 20 tunnels	Delay
[18]	TE based on actor-critic reinforcement learning and active monitoring to increase the overall service availability	Traffic engineering Active monitoring	Generic	Simulation	2 CPEs - 10 tunnels	Service availability, delay
[19]	Demonstration on dynamic path selection based on end-to-end delay active monitoring	Traffic engineering Active monitoring	Generic	Emulation: Openvswitch and VyOS	3 CPEs - 2 tunnels	Delay, packet loss
This paper	**TE for delay-sensitive services with transport based monitoring**	**Traffic engineering Passive monitoring**	**Real-time video streaming**	**Emulation: 1) Openflow based switch Aruba 2930F Real-world: 2) Municipal network**	**1) 4 CPEs - 3 tunnels 2) 2 CPEs - 2 tunnels**	**Recovery time, service availability**

requirements, elastic traffic demand and full control over the edge servers and data-center networks. SDN allows advanced centralized TE policies that allocate bandwidth among competing services based on applications priorities. In particular, they build a TE application with the aim of running their WAN links at near 100% of utilization, corresponding to 2-3x efficiency improvements relative to standard practice. Their solution enables to deploy cost-effective WAN bandwidth maximizing the network utilization.

Some academic works have built their research upon these concepts. Authors in [10] perform a techno-economic analysis about implementing SD-WAN with 4G/LTE for Automated Teller Machine (ATM) networks. Most ATMs use only the Very Small Aperture Terminal (VSAT) access to connect their WAN via satellite. Having only VSAT access on most ATMs can be risky, especially if the satellite connection goes down. With SD-WAN, ATM will have at least two WAN connections to its network; as a result, if one of the connections is down, network traffic will not be interrupted. Based on the techno-economic analysis provided by the authors, the implementation of SD-WAN with 4G/LTE for the ATM network is feasible and profitable.

The authors in [11] propose a traffic engineering optimization algorithm with the aim of minimizing traffic disruptions and the cost related to the use of different WAN access technologies, such as those based on LTE, DSL, Cable, etc. Their algorithm is supported by an active monitoring module, which consists of sending periodically end-to-end probes between CPEs to infer failures and performance degradation in the underlay WAN. The optimization algorithm is based on a minimum cost network update (Min-Cost) problem that minimizes the network reconfiguration costs. The performance evaluation is made in terms of reconfiguration cost and number of disrupted flows by considering an increasing number of network traffic demands.

In [12], the authors propose a traffic engineering algorithm based on traffic classification and prioritization. Each edge node connects to the public Internet through standard low cost access technology WAN, such as xDSL, PON, cable modems, and even LTE/5G. They consider a topology made by 3 CPEs and 5 tunnels. The authors claim that, in many cases, their proposed approach provides up to four times the end-to-end goodput of that provided by conventional traffic engineering algorithms.

The authors in [13] propose an SD-WAN solution to connect two Software Defined Data Centers (SDDCs), ensuring a predefined level of QoS and traffic prioritization. This deployment is entirely made by emulated software, such as Mininet[1] for the SDN network, Floodlight [23] as controller and vmware[2] for the virtualization of the SDDCs. The main goal of the authors is to demonstrate that a service, with a required level of QoS, such as VoIP, can be guaranteed and to provide traffic priority in an SD-WAN network. Tests are performed by generating from 150 to 300 VoIP calls and generic TCP/UDP traffic flows between the SDDCs. Results refer to bandwidth and CPU consumed by the controller that handles the connections. The authors claim that QoS in terms of minimum bandwidth, can be guaranteed in an SD-WAN solution that interconnects two SDDCs. The controller can efficiently manage 300 VoIP calls, using a maximum of 16% CPU load.

The authors in [14] propose a Dynamic Traffic Management (DTM) strategy to tackle the problem of minimizing traffic transit expenses. It refers to different monetary costs of inter-WAN domain traffic, such as the one related to the network energy consumption, and other kind of costs related to the volume of traffic. The authors focus on the optimization of costs related to traffic transfer via WANs. The ability of SD-WAN to switch traffic flows from one link to another efficiently minimizes transit expenses.

In [15] the authors propose a Cloud network architecture in which multiple data centers are connected through different public Internet Service Providers (ISPs). An overlay network is created by setting up virtual tunnels whose nodes are data centers. Concerning control plane, they are connected to a centralized SDN controller that sets forwarding rules for the

[1]Weblink (accessed on 20/11/2021): http://mininet.org/

[2]Weblink (accessed on 20/11/2021): https://www.vmware.com/

created overlay topology. Authors focus specifically on a two edge-node case, by proposing a 1:1 protection scheme with a pair of overlay tunnels, as such the two tunnels are created into two different WANs, managed by different ISPs. Traffic flows are divided into critical and non-critical, as consequence, whenever a failure occurs on a path, non-critical traffic is stopped, while critical flows are directed to the back-up path. The architecture is composed by different software modules working over the Java-based Floodlight Controller [23].

The authors in [16] present a novel implementation of Multi-Path TCP onto SD-WAN, called WAN-aware MPTCP (WaMPTCP), which optimize the WAN paths utilization by aggregating multiple (heterogeneous) WAN paths. WaMPTCP is also capable of adapting to network failures or congestion by providing fast failure recovery to applications. The authors focus on applying MPTCP to an SD-WAN scenario in order to fully exploit the available bandwidth of WANs. They implement their proposed solution into an emulated testbed made by 2 CPEs and 5 tunnels, and a real world testbed made by 3 CPEs and 2 tunnels.

The authors in [17] exploit active tomography where probes from the edge of the network are used to infer internal WAN characteristics. Specifically, the WANs are treated as black boxes where most of the networks features of interest for traffic engineering purposes are not directly observable. The work focuses on an SD-WAN solution for inter-datacenter networks in which two edge nodes distribute the traffic load among 20 different WAN connections. The authors propose a traffic engineering algorithm based on critic-only reinforcement learning (RL) by exploiting active tomography to get end-to-end delay measurement. The idea is to use these measurements for training the algorithm to distribute the traffic load among the different WAN connections. The main goal is to reduce the end-to-end traffic delay by distributing the traffic among the WAN connections.

Our previous works in [18] and [19] present an implementation of SD-WAN based on open source components, such as OpenDaylight [24] as SDN controller, OpenvSwitch [25] (OvS) and a set of services for network monitoring and policy-based path selection. In [19], we present a demo-test in a simple emulated but realistic network environment, showing new features and advantages for the enterprise in terms of resource optimization. We exploit a dynamic path selection algorithm based on end-to-end delay measurements with the aim of fast recovery from WAN failures. In [18], we propose a deep reinforcement learning for traffic engineering in SD-WAN. Considering the well known hub-and-spoke topology for enterprise networks, the work focuses on optimizing the service availability for enterprise services. In particular, we implemented an actor-critic reinforcement learning algorithm with the goal of distributing the traffic load between two edge nodes among different WAN connections (from 2 to 10). The main goal of the RL-based algorithm is to learn how to distribute the traffic to increase the service availability by keeping the end-to-end delay under a certain threshold. In particular, we design different ad-hoc reward functions to train the algorithm to avoid specific behaviours such as the WAN flipping problem due to the variation of end-to-end traffic delay.

Most of the research works presented in this section share the effort to demonstrate with simulated, emulated and real-world testbeds, the possibility of obtaining high levels of QoS even without QoS-guaranteed connectivity such as the one provided by an MPLS-based connectivity service. They focus more on the implementation and functionalities of SD-WANs with generic network traffic and less on analyzing its performance in terms of service availability and network failure resilience for specific enterprise services.

With this work, we want to fill this gap by implementing an open-source SD-WAN solution aiming at increasing the availability of delay-sensitive services, such as real-time video streaming. To do so, we developed an SD-WAN application, called SD-WAN TE, running inside the ONOS controller and a monitoring system based on eBPF, called TPM system. The goal is to manage delay-sensitive services running over the TCP transport protocol in order to achieve fast recovery from WAN failures. We validate our findings by deploying an open-source SD-WAN solution into a lab testbed and a real SD-WAN prototype. Finally, we show a performance analysis in terms of service availability and total recovery time.

Table I summarizes the current (and recalled) state of the art research works by emphasizing their novelties and differences with this paper. The next section will dive into the proposed SD-WAN TE and TPM systems design and implementation.

B. TCP-enabled passive monitoring techniques

Network monitoring techniques can be split in *active* and *passive*. The former method injects traffic probes into the network and analyzes their behaviour, while the latter does not need to inject any additional traffic into the network: statistical information is gathered by network nodes directly by observation. This method generates low or even zero overheads, however, it requires full access to the devices being monitored (e.g. routers, switches, servers, etc.). In this paper, we focus on passive network monitoring.

In traditional networks, passive monitoring mostly relies on *capture-and-analyze* tools that need expensive instrumentation and infrastructure. IPMON [26] is a passive delay measurement tool, which captures the header of each TCP/IP packet, timestamps it and sends the collected data to a central server for analysis. Another well-established passive monitoring tool is NetFlow [27]. Authors in [28] and [29] design two algorithms to use the NetFlow function of a network device to detect packet loss. However, the detection accuracy and real-time performance of these methods suffer from the fact that not all packets of traffic flows can be intercepted, as NetFlow natively performs packet sampling.

Other approaches presented by different authors in [30] [31] and [32] make use of different parameters from the TCP protocol (e.g. sequence number, ack number, etc.) to detect packet loss. However, these methods are time and space consuming due to the collection and processing of a huge amount of network traffic samples. Authors in [33] propose an alternative framework based on packet sampling performed by the routers to detect packet loss in real-time for both TCP

and UDP traffic flows. After collecting the samples, a feature extraction from TCP and UDP flows is performed and two machine learning models (i.e. Random Forest and Extreme Gradient Boosting) are trained to predict the packet loss rate.

Considering the context of wireless networks, authors in [34] propose a passive monitoring methodology called Periodic Passive Ping (PePa Ping) for Android devices. PePa Ping periodically obtains different TCP parameters such as RTT, jitter, and number of lost packets of all traffic flows. This passive approach relies on the implementation of a local VPN server residing inside the client device to collect TCP parameters directly by the Linux kernel. Authors in [35] focus on satellite communication and propose to passively monitor the retransmission rate of TCP flows to estimate the packet loss rate. In particular, they capture flowing packets and detect retransmissions by matching the TCP header fields, such as replications of the same sequence numbers.

In SDN networks and especially thanks to protocols like OpenFlow [36], monitoring has become more powerful since flow statistics, up to the transport layer, can be directly obtained from the flow tables within the switches and forwarded to monitoring collectors. Moreover, OpenFlow switches can directly report link failures.

Different from all these papers, we present a passive monitoring system able to detect lost TCP segments in the locations where the traffic flows terminate or are generated, that is, within the hosts involved in TCP sessions. Moreover, our system collects detailed information related to TCP flows that may experience packet losses, and not of all active flows (e.g. UDP flows). In the next section we will dive into the details of our proposed solution.

III. SD-WAN TE and TPM System: Overview and Implementation

The SD-WAN TE and TPM systems are designed specifically to meet the requirements of an enterprise that needs to run delay-sensitive services, based on TCP transport protocol, in their SD-WAN. Figure 1 shows a typical SD-WAN-based EN connecting branch offices to a headquarter with multiple overlay VPNs built over different access technologies WAN, such as 4G/5G, xDSL, fiber optics, etc. Each VPN is based on various overlay tunneling protocols, such as GRE[37] and VxLAN[38].

We developed an ONOS application called SD-WAN TE to manage and improve traffic engineering at the edge of the EN. The SD-WAN TE is supported by a monitoring system, called TPM, responsible for getting real-time traffic flow information from the server placed at the headquarter that runs enterprise services. For example, for each TCP flow, it can collect the number of retransmitted segments, the congestion window value, the estimated round trip time, etc. According to the required service performance constraints, SD-WAN TE is in charge of switching the tunnel in use by updating the forwarding tables of the CPEs. In a nutshell, the goal of SD-WAN TE and TPM system is to provide a quick reaction to WAN failure/congestion where CPEs perform a tunnel handoff based on information retrieved directly by the running

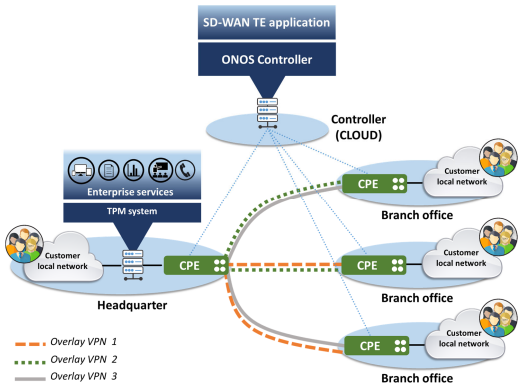

Fig. 1: SD-WAN TE and TPM system on a typical EN.

services. Next subsections will dive into the details of TPM system and SD-WAN TE.

A. Transport-layer Passive Monitoring (TPM) system

As already well known, link failures and congestion events adversely affect TCP performance [39]. Several TCP optimization schemes have been proposed to mitigate TCP throughput degradation and packet loss [40] [41]. Differently from previous research works, we developed a monitoring system able to read and store TCP flows information, such as the number of retransmissions of TCP segments, running on a server. In the SD-WAN context, the controller can only manage devices that are at the edge of the network (CPEs). Therefore, if any link or node failures occur in the WANs, the effect is reflected on the TCP traffic in the form of increasing number of retransmitted segments. Our TPM system is in charge of measuring the number of TCP retransmissions of the traffic flows and warning the network controller of possible failures on the specific WAN. The proposed TPM system, which traces

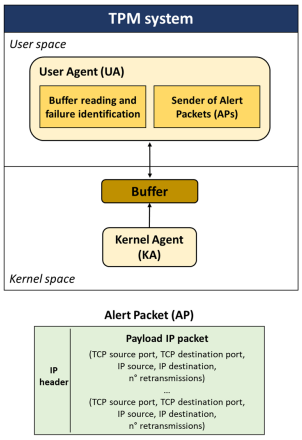

Fig. 2: SD-WAN TE and monitoring system.

TCP retransmissions on the server by means of eBPF [8], has been coded in C language and is schematically depicted in Figure 2. To this aim, we implemented a *Kernel Agent* (KA), running in the kernel space, able to collect different kinds of information from the traffic flows that are running

into the server in real-time, such as: TCP source port, TCP destination port, IP source, IP destination and number of TCP retransmissions. In a nutshell, eBPF is able to trace specific kernel functions such as those related to the operation of the TCP protocol. It means that we are able to code programs that can be automatically executed inside the kernel whenever a specific kernel function is activated. For instance, the function called `tcp_retransmit_skb` takes care of retransmitting a lost TCP segment, and whenever it is called by the kernel, our KA is executed automatically; in other words, KA is triggered every time a traffic flow experiences a TCP retransmission. Afterwards, the data collected by KA is sent to a *buffer*, which can be read by a program in the user space. A *User Agent* (UA) is in charge of processing the information collected by KA after having read the buffer. Specifically, it counts the number of TCP retransmissions per traffic flow in a given time interval and monitors whether a pre-defined threshold (TH) is exceeded. TH represents the maximum number of TCP retransmissions that can be tolerated for each traffic flow. For instance, if a service can tolerate 10 packets lost per second, then TH can be set equal to 10 directly by the user as a parameter of the UA. If a traffic flow overcomes TH, the TPM system will send out an *Alert Packet* (AP), see figure 2, containing the information of that connection, i.e., a tuple including TCP source port, TCP destination port, IP source, IP destination and number of retransmissions. The AP is crafted using Python's scapy library[3]. When the CPE receives the AP from the server, it forwards the AP towards the ONOS controller.

B. SD-WAN Traffic Engineering (SD-WAN TE) application

The goal of the SD-WAN TE application is twofold. The first is to assign the tunnel to the traffic service that requests it, while the second is to manage the dynamic re-assignment of the tunnels based on the performance of the WAN networks. When the application receives a network demand, it assigns the first free tunnel or, if it does not exist, creates a new one. Moreover, every time the TPM system detects a service degradation due to many TCP segments retransmissions, it informs the SD-WAN TE by sending an Alert Packet (AP) containing the information regarding the degraded service, as seen in the previous subsection. Figure 3 shows the SD-WAN TE application modules.

When an AP is received, it is processed by the *AP classification module*, which reads the packet payload and classifies which services are degraded. This operation is performed thanks to a Quality-based routing (QbR) table which sets the maximum number of TCP retransmissions a service can tolerate. The purpose of this table is to define thresholds on network parameters that cannot be exceeded by application services. In particular, the QbR table contains an entry for each type of service identified by: TCP source port (`TCP src`), TCP destination port (`TCP dst`), IP source (`IP src`), IP destination (`IP dst`), service and QbR Threshold (`QbR TH`), as shown in figure 3. In this paper, we assume a software module capable of filling this table. Given the goal of this paper, the development of this software module is out of the scope, so we assume a pre-filled QbR table.

[3]Weblink (accessed on 12/11/2021): https://scapy.net/

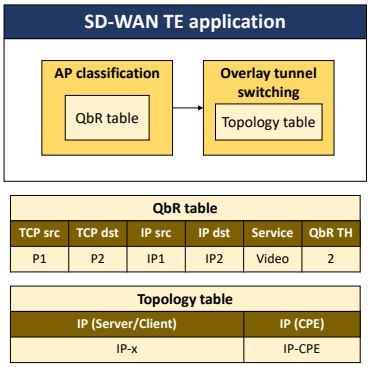

Fig. 3: SD-WAN TE application modules. The QbR and Topology tables show a typical example of an entry line with information regarding the details of traffic flows and their sources and destinations.

Once the degraded services have been identified, the *overlay tunnel switching module* has the task of changing the tunnel used by the degraded services to another. This module contains a topology table that maps the IP address of the clients with that of the CPEs to which they are connected, and is updated through the control messages of the Openflow 1.3 protocol. The tunnel switching is made on the basis of the health of the tunnel in terms of (i) number of services currently flowing and (ii) current number of retransmitted TCP segments. Specifically, the health of the tunnel is evaluated by tracking the number of TCP retransmissions of each service flowing on the tunnel. For example, considering a specific application service, if its TCP retransmissions exceed the respective QbR threshold, the service will be routed to the first tunnel where the number of TCP retransmissions of the current services is less than the respective QbR thresholds. The tunnel change happens by updating the forwarding table of the CPEs.

The proposed overall SD-WAN TE and TPM system working procedure is shown in Algorithm 1.

By default, TH is set to 1 by the UA. This means that, every time a retransmission occurs, KA collects and writes to the buffer the TCP traffic flows information. The UA reads the buffer and counts the number of TCP retransmissions occurred in a given time interval, which can be dynamically chosen. Then, if the TH is exceeded by one or more TCP traffic flows, the UA sends an AP to the CPE. The CPE sends this AP to the ONOS controller and then the SD-WAN TE application reads the payload of the AP. Finally, if one or more TCP flows overcome the QbR TH, the controller triggers the tunnel change by updating the forwarding tables of the CPEs.

C. Comparisons with traditional passive monitoring

As mentioned in the previous sections, the TPM system is based on passive monitoring, and it differs from traditional passive monitoring techniques (see Section II-B and [42]) mainly on implementation and on the way TCP flows are

Algorithm 1: SD-WAN TE and TPM system algorithm

KA is triggered when the `tcp_retransmit_skb` function is activated;
KA retrieves the tuple $t_i \in T$ for each TCP active flow i and submit it to the buffer.
$t_i : [TCPsrc, TCPdst, IPsrc, IPdst, N_{retrans}]$;
UA reads the buffer and store the TCP flows info according to a threshold TH as follows;
for $t \in T$ **do**
    if $t[N_{retrans}] \geq TH$ **then**
        Store t into a list L;
    end
end
UA sends an AP to the controller containing L into the payload;
SD-WAN TE receives the AP, reads the payload and extracts L;
SD-WAN TE classifies the TCP flows by performing a linear search on the QbR table;
for *each TCP flow with* $N_{retrans} \geq$ *the corresponding QbR TH* **do**
    Switch/Assign the tunnel for that TCP flow with the first available tunnel that contains traffic flows whose number of current TCP restransmissions is $\leq$ than the corresponding QbR TH;
end

monitored. Below, we present differences and similarities with traditional passive monitoring techniques, whose features are summarized in Table II:

1) TPM requires less CPU usage than traditional methods, especially when no TCP retransmissions occur. Traditional methods consume many CPU cycles as they sample and process traffic to and from network devices.
2) TPM has been developed to monitor TCP-only connections, while traditional methods can monitor any kind of traffic. However, TPM functionalities could be extended to cover a larger variety of traffic.
3) TPM can detect TCP segment losses in real time as it is installed where connections are generated/terminated. Instead, traditional methods are not suitable for real-time detection of losses as all traffic must be sampled and analyzed, which is a time-consuming task.
4) TPM requires a software agent to be installed on server/client machines and is only compatible with Linux operating systems (kernel $\geq 4.x$). Conversely, traditional methods require the installation of software inside the CPEs and can take advantage of embedded protocols that must be supported (e.g. NetFlow, Bidirectional Forwarding Detection, etc.).
5) TPM analyzes all TCP traffic flows, even non-SD-WAN. While traditional methods can carry out a preventive filtering of the type of traffic to be analyzed.

A customer using SD-WAN should be aware of the pros and cons in Table II to make the most appropriate choice on the monitoring system to be adopted by its SD-WAN solution. Furthermore, since the customer usually owns most of the IT equipment needed to provide its services to end users, it can easily implement the TPM system in its servers so that high-priority TCP connections can be controlled in real-time. Additionally, it must also take into account the limitations of this approach and may decide to complement it with other additional monitoring strategies. For instance, on the same SD-WAN network the customer can also implement (i) traditional passive monitoring methods operating within the CPEs and/or (ii) active monitoring techniques to make up for the shortcomings of passive methods (see Section II-B).

IV. Testbeds description

In this work we developed two SD-WAN testbeds. The first one, shown in figure 4, called *Lab* testbed, is a controlled implementation of an SD-WAN. The objective is to use this controlled testbed to develop and study traffic engineering and monitoring algorithms in order to test them on real SD-WANs. The second testbed, shown in figure 5, called *Municipal*, is the result of a collaboration between Politecnico di Milano, SWAN networks (a university spin-off company of the Politecnico di Milano) and the municipal administration of the Italian city of Militello in Val di Catania (V.C.). The goal is to test the algorithms developed in our lab on a live SD-WAN implementation to evaluate performance and limitations. We deployed our SD-WAN solution made by SD-WAN TE + TPM system on both testbeds to validate it (see section V).

A. Lab testbed

Considering figure 4, the Lab testbed is made up of four main elements:

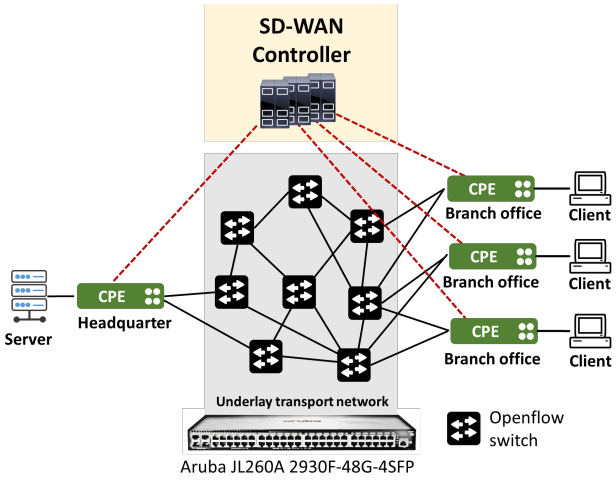

Fig. 4: SD-WAN testbed in our lab at Politecnico di Milano.

- *SD-WAN controller*: we used the open-source ONOS [9] SDN controller, which is responsible for managing the CPEs through the SD-WAN TE application presented in the previous section. We instantiated the controller on a server located in our lab at Politecnico di Milano (Milan, Italy) with a Linux operating system.

TABLE II: Comparisons between traditional passive monitoring techniques and TPM system

	Traditional Passive Monitoring techniques	TPM system
1	High CPU usage	Low CPU usage
2	Any kind of traffic	TCP-only
3	No real-time detection of segment losses	Real-time detection of segment losses
4	Software to be installed on the CPEs	Software to be installed on the server/client machines (Linux-only)
5	Easy connections filtering and isolation	Additional overhead for non-SD-WAN connections

- *Underlay transport network*: we used an Openflow-enabled switch (Aruba JL260A 2930F-48G-4SFP) that is capable of creating several emulated instances of Openflow switches by statically binding its physical ports to each instance. We emulated 8 Openflow switches, see figure 4, acting as internal WAN devices managed by an Internet Service Provider (ISP), therefore they are not controllable by the SD-WAN controller. For this reason, they are connected to a different ONOS instance that we do not show in the figure for clarity. In any case, the control over the internal WAN devices allows us to test the limitations and performance of an SD-WAN. For example, we can emulate controlled congestion and failure events.
- *CPEs*: they are made by Raspberry Pi computers (model 3B+ with Raspbian as operative system) in which we have installed OpenvSwitch (version 2.12.0). The latter was exploited for the switching functionality and for the establishment of the overlay tunnels. Specifically, we applied the Generic Routing Encapsulation (GRE) tunneling protocol [37] to implement different overlay tunnels.
- *Headquarter server*: it is a server where we have installed and developed the TPM system. It consists of 8 Intel Xeon processors with 128 GB of RAM in which we have installed Ubuntu server 20.04 LTS and eBPF. This server is used as the source of enterprise services to be monitored via the TPM, such as real-time video streaming flows. The clients that request the services are represented by three PCs placed at the branch offices.

B. Municipal testbed

The Municipal testbed is made by two CPEs and two overlay tunnels based on GRE tunneling protocol. The city-hall of Militello is connected to a remote branch office of the municipality by two different networks. In each one of the two networks a tunnel has been created to connect the two sites. The aim of the SD-WAN solution is to improve network availability by dynamically switching the inter-site traffic flows between the two tunnels. The switching occurs based on the status of the traffic flows running into the two tunnels, constantly monitored by means of the proposed TPM system.

We present the physical architecture of the testbed in figure 5. There are two CPEs in the data layer, based on Raspberry Pi as in the Lab testbed, one at the city hall and the other at the remote office, connecting the hosts placed in the two administrative offices to the WAN tunnels. One of the two WAN networks interconnecting the CPEs is a private WLAN owned by the municipality of Militello V.C., while the second

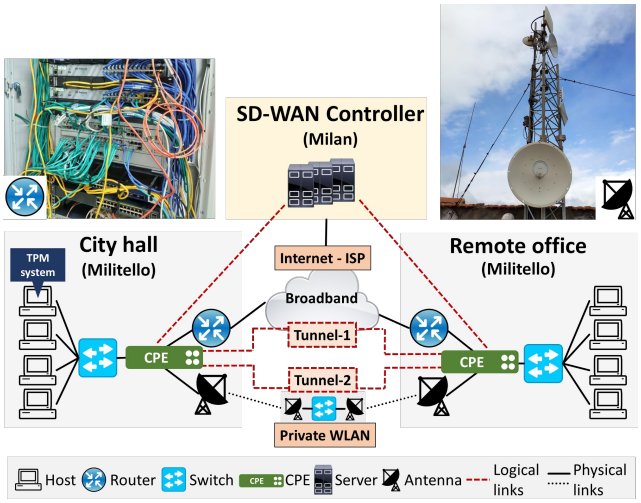

Fig. 5: SD-WAN testbed in Militello V.C.

is an Internet/Broadband network provided by an Italian ISP. As said, we applied the GRE tunneling protocol [37] to implement the two tunnels, see Tunnel-1 and Tunnel-2 in figure 5, between the CPEs through the two networks. To implement the SD-WAN controller we used the same ONOS controller as in the Lab testbed. Originally, the two sites were connected only by the WLAN, which is owned by the municipality and is therefore free of charge. However, this network is often subject to failures due to hardware problems and bad weather, which causes small displacements of the antennas from their optimal position. So, the municipality decided to contract the ISP to switch to an interconnection through fixed broadband access, which is on the opposite a paid service. The idea underlying our testbed is, by SD-WAN, to enable a mechanism that normally routes traffic on the free-of-charge WLAN connection, switching to the ISP network only as a backup. In this way, we can improve the availability performance of the inter-site connection, while minimizing the usage of the paid ISP network, thus reducing the cost for the municipality. Our SD-WAN solution is able to manage the traffic flows between these two networks.

V. Experiments and Results

In order to measure the performance of the proposed SD-WAN solution, we conducted different experiments on both testbeds and measured two performance metrics, that are: total *Recovery Time* (RT) and *Service Availability* (SA). We run multiple TCP flows between the CPE at the Headquarter and the CPEs at the branch offices by means of D-ITG [43] tool. We assume those TCP flows being part of delay-sensitive

services, so with very narrow QoS thresholds, therefore, we impose a TH on the number of retransmissions less or equal to 10. We evaluated the performance of our solution by considering two types of network failures that can generate packet loss: congestion events, emulated on both testbeds, and WAN link failures, emulated only on the lab testbed. The former have been generated by inducing packet losses on the overlay tunnels of both testbeds by means of NetEm [44], a tool able to emulate the properties of WANs such as variable delay, packet loss, duplication and re-ordering. The latter was carried out by manually disconnecting the links of the emulated WAN network testbed within our laboratory at the Politecnico di Milano.

A. Evaluation of the TPM system under network congestion events

1) Lab testbed: This section evaluates the total Recovery Time (RT) and the Service Availability (SA) by considering the Lab testbed in figure 4 under network congestion events. Table III shows the technical setup of the Lab testbed.

TABLE III: Lab testbed setup

Lab testbed	
Threshold (TH)	1, 3, 5, 10
Packet Loss (PL)	1%, 5%, 10%
Congestion events	uniformly distributed from 100 to 200
Average duration of the congestion event	5 s
Number of TCP flows	from 50 to 150
Topology	4 CPE (1 Headquarter and 3 branch offices); 3 overlay GRE tunnels
Single experiment duration	2 hours
Number of experiments	50

Figure 6 shows the average RT by considering different Packet Loss (PL) percentages and Thresholds (TH). In particular, we look at the following time intervals:

- Detection Time (DT): average time interval elapsed from the start of the congestion event to its detection by the TPM system;
- Switch Path Time (SPT): average time interval elapsed from the sending of the AP to the path switching:
- Total Recovery Time (RT): the sum of the two previous time intervals.

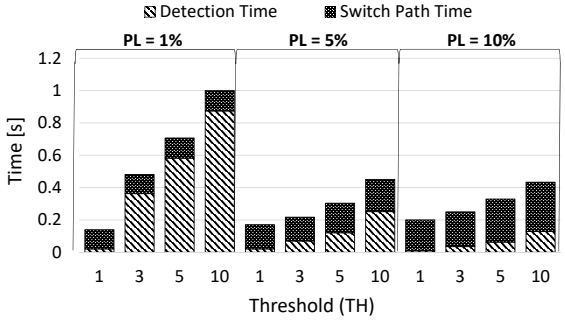

Fig. 6: Total recovery time with different PL and TH values.

The DT is inversely proportional with respect to the increase of PL, while the SPT increases together with PL. Since the packet loss is set to the network interfaces of the CPE, the control plane traffic is also affected, consequently increasing the percentage of packet loss also increases the SPT. Indeed, when the PL is small, it means that few TCP connections are experiencing retransmissions, so the traffic switching procedure takes less time. On the other hand, when the PL increases it means that many TCP connections are experiencing retransmissions, as such, the time to process and to make the tunnel change is higher. In particular, figure 7 shows the probability density function of DT and SPT when TH=1 and PL=5%. We can see that the total recovery time mostly depends on the SPT. As a result, optimizing the SPT procedure means reducing the overall RT.

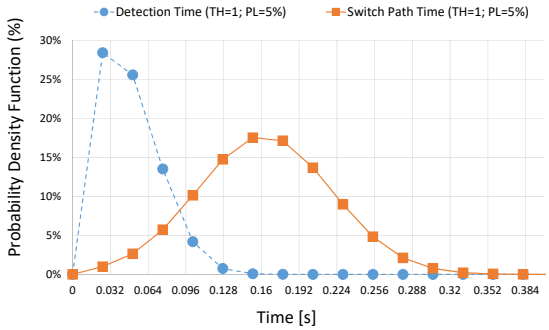

Fig. 7: Probability density function of DT and SPT when PL=5% and TH=1.

Considering a PL=5% and TH=10, figure 8 shows a detailed perspective on the result in terms of bitrate and delay of a single TCP flow. In the figure, the start and end of the congestion can be graphically identified by the drop and rise of the bitrate and by the rise and drop of the delay, respectively. The total RT span between these two time instants is around 400 ms. The vertical red line indicates the time of the 11^{th} retransmission, that triggers the AP. In this experiment the total number of retransmissions stopped at 34, right before the switch to the backup tunnel. In particular, as TCP retransmissions increase, the instantaneous delay of the flow also increases, which returns to a low value within a few hundred of milli-seconds. However, in this time interval, i.e. the RT, the instantaneous delay gradually increases and then returns to acceptable values. This mechanism helps to keep the delay value very low on average. If we did not apply this mechanism, the delay value would continue to grow, significantly affecting the quality of the service.

Figure 9 shows the performance of the proposed implementation in terms of Service Availability with different values of TH and PL. Specifically, we define SA as percent uptime, i.e., the total time in which TCP traffic is routed into the uncongested tunnel; in other words, the total time in which the number of retransmissions remained below the QbR TH.

We can notice that the SA decreases as the TH increases for each value of PL. When we increase the value of TH, the total RT increases, as a result, the downtime also increases by affecting the overall SA. Furthermore, we can note that the SA when PL=5% is slightly higher than when PL=10%. This is due to the fact that, by increasing the PL, we also increase the

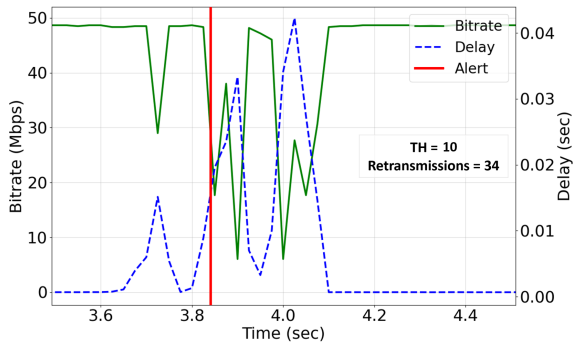

Fig. 8: Evolution of a TCP flow's Bitrate and Delay.

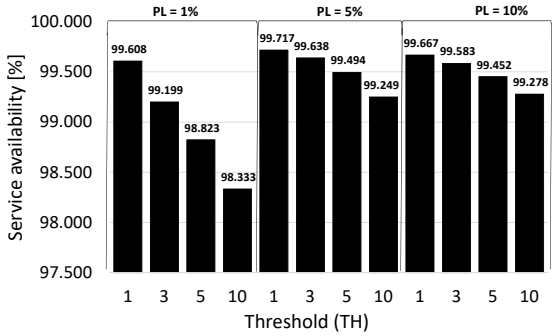

Fig. 9: Service availability with different TH and PL values.

number of lost packets of the control plane traffic responsible for the tunnel switching in the CPEs. In fact, this is visible from figure 6, where the SPT increases as the PL increases, elevating the total RT. If, on the other hand, we do not use the proposed SD-WAN solution, the maximum reachable SA is 91.666%. As a result, our solution increases total SA by at least 6.667% to a maximum of 7.102%.

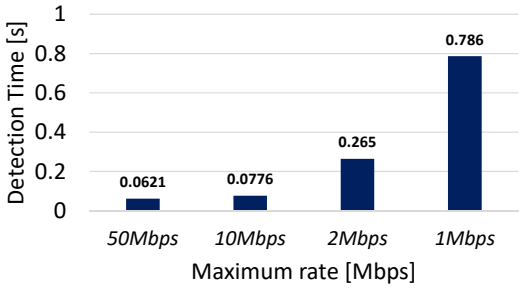

Fig. 10: Detection time with different maximum rates imposed on the tunnels.

Figure 10 shows the performance of the proposed TPM system in terms of DT with different bandwidth limitations for the monitored application services. In particular, we used TBF qdisc[4], a tool to shape the traffic at different maximum rates, and with {PL=10%; TH=5; 10 TCP flows}. This experiment wants to emphasize a classic characteristic of passive monitoring, that is, the DT of the retransmitted segments depends on the amount of traffic of the monitored application service. We can see from figure 10 how the DT and the maximum rate are inversely proportional. This aspect has an effect on the monitored application services with low traffic rates, such as tens of Kbps or a few Mbps. Being the DT higher, the SD-WAN TE takes longer to reroute the services.

2) Municipal testbed: This section evaluates the total Recovery Time and the Service Availability by considering the Municipal testbed in figure 5. Table IV shows the technical setup of the testbed. In particular, we were able to test our SD-WAN solution during three working days of the Municipality administration, as such, we could run a subset of the experiments that we run into the Lab testbed. We considered a TH equal to 10 and we could run at most 50 TCP flows.

TABLE IV: Municipal testbed setup

Municipal testbed	
Threshold (TH)	10
Packet Loss (PL)	1%, 5%, 10%
Congestion events	uniformly distributed from 100 to 200
Average duration of the congestion event	5 s
Number of TCP flows	from 10 to 50
Topology	2 CPE, 2 overlay GRE tunnels
Single experiment duration	8 hours
Number of experiments	3

Table V shows the average RT and SA for the Municipal testbed. In this case the DT and the SPT are higher than those measured in the Lab testbed. It depends on the geographical distance between the CPE and the SD-WAN controller, in fact the CPEs and the controller are located at a distance of 1000 km, as such, it directly affects the SA. This result leads us to conclude the importance of the controller location for those services where immediate controller intervention is required. In order to minimize the DT and the SPT, we could apply various re-routing strategies based on proactive methods, such as instructing before hand the CPEs which tunnels to use in case some TCP flows should experience retransmissions. The latter is left for future work.

TABLE V: Municipal testbed: recovery time and service availability evaluation

	DT	SPT	RT	SA
Packet Loss 1%	0.962 s	0.151 s	1.113 s	99.033 %
Packet Loss 5%	0.645 s	0.207 s	0.852 s	99.260 %
Packet Loss 10%	0,386 s	0.238 s	0.624 s	99.458 %

B. Evaluation of the TPM system under WAN link failures

This section evaluates the RT and SA by considering the Lab testbed in figure 4 under WAN link failures. We compared our SD-WAN solution based on the TPM system with a baseline tunnel handoff procedure based on the Bidirectional Forwarding Detection (BFD) protocol [45]. BFD is a network protocol that is used to detect link failures between two connected routers or switches. It is integrated in OvS and used for detecting link failures. By default, BFD declares a link failure after three failed handshakes and declares a link recovered after one successful handshake. Each BFD packet

[4]Website: https://man7.org/linux/man-pages/man8/tc-tbf.8.html

can be sent to BFD peers within a minimum of 50 ms, as a result, BFD can detect a fault only after 150 ms. However, we need to take into consideration the processing time to elaborate the failure and inform the controller. This additional time is strictly related to the testbed setup.

For this experiment, we compared how our SD-WAN solution reacts to WAN link failures by physically disconnecting the links in the lab during the experiments. In particular, we focused on analyzing the RT for both approaches, namely SD-WAN based on TPM and on BFD. This type of experiment shows that, after a link cut, the number of retransmissions generated by the TCP protocol is two (at most); consequently, in order to capture the link cuts, we set the TH to 1 and 2. Table VI shows the results of this set of experiments. Considering the setup of the Lab testbed, our solution obtained an average value of RT equal to 0.876 s (TH=1) and 1.4 s (TH=2), while the one based on BFD obtained an average value of 0.340 s. In this case, the monitoring approach based on BFD protocol highlights how our solution is less suitable for the detection of WAN link failures.

This experiment also makes a fundamental tradeoff emerge, that is, which is the best choice of the TH. A low TH readily responds to a possible network problem but increases the likelihood of numerous unnecessary handoffs making the overall SD-WAN unstable. On the other hand, a high TH reduces the likelihood of erroneous handoffs but increases the service unavailability. Clearly, the number of TCP retransmissions as the only metric for determining network failures is shown not to be enough, and makes important to investigate other innovative techniques so that the broadest types of failures can be embraced. As mentioned in section III-C, a winning approach could be combining various monitoring techniques that are complementary to each other and that are able to promptly prevent the different types of failures that the network may incur.

TABLE VI: Comparison between SD-WAN based on TPM and BFD.

		RT	**SA**
SD-WAN based on TPM	TH=1	0.876 s	99.2325 %
	TH=2	1.4 s	98.7847 %
BFD		0.463 s	99.5467 %

VI. Discussion on open issues and future work

In this paper, we have presented an SD-WAN solution optimized for high-priority enterprise application services. We must point out that based on the choice of TH and QbR TH parameters, other types of services that are not high-priority can also be optimized. For instance, low-priority application services could be handled through load balancing techniques based on the amount of traffic flowing on the tunnels. We decided to focus on delay-sensitive services due to their popularity and importance in a 5G network context. Nevertheless, the inclusion of low-priority traffic in our testbed will be considered in our future work. We have demonstrated how an eBPF-based monitoring system implemented at the transport layer can be reliable and performing. However, there are some problems and limitations of our solution to consider. Our proposed SD-WAN solution is limited to applications that use TCP as the transport protocol. This represents a limitation of our system, but at the same time an interesting starting point for a future work. For instance, a possible extension of the TPM system is to consider other transport protocols such as UDP to optimize those application services that make use of it. eBPF may not be supported by all server operating systems on the market. On the other hand, recent developments show that this tool is going to be supported by multiple operating systems, which makes the applications that use it supported[5]. The QbR table dimension shown in figure 3 could pose a scalability problem in terms of number of entries. Although the filling of this table is out of the scope in this paper, it represents an interesting future work. That is, the implementation of a system able to recognize the type of service in real time. Moreover, in this paper we have focused on effectively managing two types of network failures that are related to sudden congestion and link cut events. However, in our future work, we will consider other failure scenarios related to the network controller, CPE, and headquarter server. These three aspects represent the starting point for our subsequent work and we believe that this solution will soon accommodate other types of services with different QoS constraints.

VII. Conclusions

An enterprise WAN is a network that connects geographically-spread sites of a company that could be located anywhere in the world. MPLS has been so far the main WAN technology for enterprise networking because of its high performance. Although MPLS has many advantages, SD-WAN is a new and fast growing paradigm that could achieve similar performance, but more cost-effectively. In this paper, we evaluated the performance of an experimental SD-WAN solution deployed in two real testbeds to deliver delay-sensitive service flows with certain QoS thresholds in the case of congestions. We have implemented a traffic engineering application directly inside the ONOS controller that operates together with a monitoring system. The latter is able to collect transport protocol information such as TCP retransmissions. We have observed the advantages of our SD-WAN solution in terms of recovery time and service availability, showing how this solution can provide high performance. In an increasingly Cloud-centric world, this revolutionary technology is universally acclaimed as a new and unprecedented way to easily implement policies across large WANs at a fraction of the cost of traditional solutions.

Acknowledgment

The authors would like to thank the administration of the Municipality of Militello in Val di Catania. A heartfelt thanks goes to the former Head of the IT Office Mario Troia, whose help and support were of paramount importance. Thank you for everything you have done.

[5]Weblink (accessed on 17/11/2021): https://ebpf.io/

REFERENCES

[1] S. Troia, M. Mazzara, L. Zorello, and G. Maier, "Performance evaluation of overlay networking for delay-sensitive services in sd-wan," in *IEEE International Mediterranean Conference on Communications and Networking (MeditCom)*, 2021.

[2] C. N. Academy, Ed., *Connecting Networks Companion Guide*. Cisco Press., 2014.

[3] R. K. Rangan, "Trends in sd-wan and SDN," *CSI Transactions on ICT 8.1*, 2020.

[4] C. Evans, J. Issa, and S. Forrest, "The sustainable future of video entertainment, from creation to consumption," in *White paper, Futuresource and Interdigital*, 2020.

[5] RFC 7540 - hypertext transfer protocol version 2 (http/2) standard. [Online]. Available: https://datatracker.ietf.org/doc/rfc7540/

[6] Z. Xu, X. Zhang, and Z. Guo, "Qoe-driven adaptive k-push for http/2 live streaming," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 6, pp. 1781–1794, 2019.

[7] R. Huysegems, J. Van Der Hooft, T. Bostoen, P. Rondao Alface, S. Petrangeli, T. Wauters, and F. De Turck, "Http/2-based methods to improve the live experience of adaptive streaming," in *Proceedings of the 23rd ACM international conference on Multimedia*, 2015, pp. 541–550.

[8] ebpf. [Online]. Available: https://ebpf.io/

[9] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow *et al.*, "Onos: towards an open, distributed SDN os," in *Proceedings of the third workshop on Hot topics in software defined networking*, 2014, pp. 1–6.

[10] S. Andromeda and D. Gunawan, "Techno-economic analysis from implementing sd-wan with 4g/lte, a case study in xyz company," in *2020 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, 2020, pp. 345–351.

[11] D. Zad Tootaghaj, F. Ahmed, P. Sharma, and M. Yannakakis, "Homa: An efficient topology and route management approach in sd-wan overlays," in *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, 2020, pp. 2351–2360.

[12] S. Lee, K.-Y. Chan, and T.-Y. Chen, "Design and implementation of an sd-wan vpn system to support multipath and multi-wan-hop routing in the public internet," in *TechRxiv*, 2020.

[13] R. E. Mora-Huiracocha, P. L. Gallegos-Segovia, P. E. Vintimilla-Tapia, J. F. Bravo-Torres, E. J. Cedillo-Elias, and V. M. Larios-Rosillo, "Implementation of a sd-wan for the interconnection of two software defined data centers," in *2019 IEEE Colombian Conference on Communications and Computing (COLCOM)*, 2019, pp. 1–6.

[14] Z. Duliski, R. Stankiewicz, G. Rzym, and P. Wydrych, "Dynamic traffic management for sd-wan inter-cloud communication," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 7, pp. 1335–1351, 2020.

[15] K. Phemius and M. Bouet, "Implementing openflow-based resilient network services," in *2012 IEEE 1st International Conference on Cloud Networking (CLOUDNET)*, 2012, pp. 212–214.

[16] Y. Zhang, J. Tourrilhes, Z.-L. Zhang, and P. Sharma, "Improving sd-wan resilience: From vertical handoff to wan-aware MPTCP," *IEEE Transactions on Network and Service Management*, vol. 18, no. 1, pp. 347–361, 2021.

[17] S. Xu, M. Kodialam, T. V. Lakshman, and S. S. Panwar, "Tomography based learning for load distribution through opaque networks," 2020.

[18] S. Troia, F. Sapienza, L. Var, and G. Maier, "On deep reinforcement learning for traffic engineering in sd-wan," *IEEE Journal on Selected Areas in Communications*, vol. 39, no. 7, pp. 2198–2212, 2021.

[19] S. Troia, L. M. M. Zorello, A. J. Maralit, and G. Maier, "Sd-wan: An open-source implementation for enterprise networking services," in *2020 22nd International Conference on Transparent Optical Networks (ICTON)*, 2020, pp. 1–4.

[20] (2020) Comparison of the sd-wan vendor solutions. [online]. [Online]. Available: https://www.netmanias.com/en/post/oneshot/12481/sd-wansdn-nfv/comparison-of-the-sd-wan-vendor-solutions

[21] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, "B4: Experience with a globally-deployed software defined wan," *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4, pp. 3–14, 2013.

[22] C.-Y. Hong, S. Mandal, M. Al-Fares, M. Zhu, R. Alimi, C. Bhagat, S. Jain, J. Kaimal, S. Liang, K. Mendelev *et al.*, "B4 and after: managing hierarchy, partitioning, and asymmetry for availability and scale in google's software-defined wan," in *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, 2018, pp. 74–87.

[23] (2013) Floodlight. [Online]. Available: http://www.projectfloodlight.org/floodlight

[24] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven SDN controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*. IEEE, 2014, pp. 1–6.

[25] Ovsdb. [Online]. Available: http://docs.openvswitch.org/en/latest/ref/ovsdb.7/

[26] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S. C. Diot, "Packet-level traffic measurements from the sprint ip backbone," *IEEE network*, vol. 17, no. 6, pp. 6–16, 2003.

[27] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes, "Cisco systems netflow services export version 9," 2004.

[28] R. Liu, S. Yang, Q. Zhang, and X. Li, "Icmp netflow records based packet loss rate estimation," in *2018 Eighth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*. IEEE, 2018, pp. 1238–1241.

[29] R. Hofstede, P. Čeleda, B. Trammell, I. Drago, R. Sadre, A. Sperotto, and A. Pras, "Flow monitoring explained: From packet capture to data analysis with netflow and ipfix," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2037–2064, 2014.

[30] Y. Gu, L. Breslau, N. Duffield, and S. Sen, "On passive one-way loss measurements using sampled flow statistics," in *IEEE INFOCOM 2009*. IEEE, 2009, pp. 2946–2950.

[31] S. Jaiswal, G. Iannaccone, C. Diot, J. Kurose, and D. Towsley, "Measurement and classification of out-of-sequence packets in a tier-1 ip backbone," *IEEE/ACM Transactions on networking*, vol. 15, no. 1, pp. 54–66, 2007.

[32] M. Mellia, M. Meo, L. Muscariello, and D. Rossi, "Passive analysis of TCP anomalies," *Computer Networks*, vol. 52, no. 14, pp. 2663–2676, 2008.

[33] H. Wu, Y. Liu, G. Cheng, and X. Hu, "Real-time packet loss detection for TCP and UDP based on feature-sketch," in *IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2021, pp. 1–6.

[34] D. Madariaga, L. Torrealba, J. Madariaga, J. Bustos-Jiménez, and B. Bustos, "Pepa ping dataset: Comprehensive contextualization of periodic passive ping in wireless networks," in *Proceedings of the 12th ACM Multimedia Systems Conference*, ser. MMSys '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 274280. [Online]. Available: https://doi.org/10.1145/3458305.3478456

[35] K. Koide, S. Fujieda, K. Cho, and N. Shiratori, "TCP retransmission monitoring and configuration tuning on ai3 satellite link," in *Technologies for Advanced Heterogeneous Networks*, K. Cho and P. Jacquet, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 282–295.

[36] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[37] RFC 2784 - generic routing encapsulation (gre) - ietf tools. [Online]. Available: https://tools.ietf.org/html/rfc2784

[38] RFC 7348 - virtual extensible local area network (vxlan): A framework for overlaying virtualized layer 2 networks over layer 3 networks. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc7348

[39] U. Ranadive and D. Medhi, "Some observations on the effect of route fluctuation and network link failure on TCP," in *Proceedings Tenth International Conference on Computer Communications and Networks (Cat. No.01EX495)*, 2001, pp. 460–467.

[40] R. Crpa, M. D. de Assuno, O. Glck, L. Lefvre, and J.-C. Mignot, "Evaluating the impact of SDN-induced frequent route changes on TCP flows," in *2017 13th International Conference on Network and Service Management (CNSM)*, 2017, pp. 1–9.

[41] G. Li and P. Jin, "A dynamically adjusted congestion control algorithm for TCP," *J. Inf. Comput. Sci.*, vol. 9, pp. 4691–4697, 2012.

[42] P.-W. Tsai, C.-W. Tsai, C.-W. Hsu, and C.-S. Yang, "Network monitoring in software-defined networking: A review," *IEEE Systems Journal*, vol. 12, no. 4, pp. 3958–3969, 2018.

[43] A. Botta, A. Dainotti, and A. Pescapè, "A tool for the generation of realistic network workload for emerging networking scenarios," *Computer Networks*, vol. 56, no. 15, pp. 3531–3547, 2012.

[44] tc-netem - linux. [Online]. Available: http://man7.org/linux/man-pages/man8/tc-netem.8.html

[45] RFC 5880 - bidirectional forwarding detection. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc5880