Extending P4 In-Band Telemetry to User Equipment for Latency and Localization-aware Autonomous Networking with AI Forecasting

DAVIDE SCANO[1], FRANCESCO PAOLUCCI[2], KOTESWARARAO KONDEPU[3], ANDREA SGAMBELLURI[1], LUCA VALCARENGHI[1], AND FILIPPO CUGINI[2]

[1] Scuola Superore Sant'Anna, Pisa, Italy
[2] CNIT, Pisa, Italy
[3] IIT Dharwad, Dharwad, India
[*] Corresponding author: davide.scano@santannapisa.it

Compiled August 16, 2021

In Beyond-5G Networks, detailed end-to-end monitoring of specific application traffic will be required along with the access-backhaul-cloud continuum to enable low latency service thanks to local edge steering. Current monitoring solutions are confined to specific network segments. In-band telemetry (INT) technologies for SDN programmable data planes based on the P4 language are effective in the backhaul network segment, although limited to inter-switch latency, therefore link latencies including wireless and optical segments are excluded from INT monitoring. Moreover, information such as user equipment geolocation information would allow detailed mobility monitoring and improved cloud-edge steering policies. However, the synchronization between latency and location information, typically provided by different platforms, is hard to achieve with current monitoring systems. In this paper P4-based INT is proposed to be thoroughly extended involving the user equipment (UE). The novel INT mechanism is designed to provide synchronized and accurate end-to-end latency and geolocation information, enabling decentralized steering policies, i.e. involving the UE and selected switches, without the SDN controller intervention. The proposal also includes an Artifical Intelligence (AI)-assisted forecast system able to predict latency and geolocation in advance and trigger faster edge steering.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

1. INTRODUCTION

The converged packet optical infrastructure that ranges from the cloud to the edge computing resources is rapidly evolving to support latency-critical 5G applications [1, 2]. New monitoring technologies have emerged to provide accurate network awareness to the control and orchestration system. A first example is telemetry [3–5], enabling network nodes to stream performance parameters such as bit error rate (BER) and optical power levels even every few seconds, at a pace significantly higher than traditional solutions e.g. based on Simple Network Management Protocol (SNMP). A second example is the P4-based In-band Network Telemetry (INT) which enables the monitoring of the actual edge-cloud experienced latency [6–9]. INT relies on extra packet headers used to carry metadata information retrieved from the forwarding plane. For example, INT can be used to collect the time spent in queue within each traversed node from the edge to the cloud by every packet of a specific 5G service [10, 11]. Then, the collected edge-to-cloud latency can feed the Software Defined Network (SDN) Control enabling re-optimization of traffic engineering and Quality of Service (QoS) solutions. In case of critical service conditions, the SDN Controller (/Orchestrator) is typically involved to recover adequate service conditions [12, 13].

Beyond 5G (B5G) is now calling for a fully end-to-end (e2e) architecture, where the infrastructure is extended from the cloud, through the edge, and up to the B5G user equipment (UE) [14]. However, monitoring actual applications performance parameters (e.g. latency) across the whole e2e path is often not feasible given the involvement of multiple service providers including the 5G Operators (particularly in case of roaming), edge and cloud providers, and transport network operator(s). Indeed, it is important to highlight that tools like ping just provide external (out-of-band) measurements and can not be used to measure the actual latency and queuing delay experienced by the application/service packets. Furthermore, retrieving and exploiting other relevant parameters such as the localization of the UE might be difficult even exploiting the latest 5G technologies

(e.g. beam-forming [15]). Indeed, such information might not be available to all involved service providers (e.g. unknown to the edge/cloud service provider). Moreover, synchronizing each generated packet with its localization information appears extremely difficult.

B5G solutions are also expected to react much faster to network events affecting latency or QoS, such as failures or congestion. However, the intervention of the SDN Controller introduces control plane delays, typically increased due to scalability issues experienced during network transients (e.g. multiple network elements simultaneously requiring re-configurations) [16, 17]. Thus, B5G aims at providing selected decentralized capabilities, such as self-management of (pre-planned) services, thus avoiding the SDN controller intervention during critical events while minimizing control overhead.

Finally, B5G is expected to provide prediction of critical events, by leveraging on Machine Learning (ML) and Artificial Intelligence (AI) techniques. Several valuable works have recently demonstrated prediction capabilities in the context of optical networking [18–22] and service level management [23–26].

In this work we first propose to extend P4 INT up to the 5G user equipment (UE). In particular, packet flows carrying selected latency-sensitive services are proposed to encompass INT header already from the UE. This enables accurate latency monitoring of the whole e2e path, including both the wireless and wired segments, even if operated by different service providers.

Second, we propose to include within the INT header also localization information. This provides the capability to understand and predict mobility conditions. It is important to note that leveraging INT for localization information guarantees accurate synchronization between service packets generated by the UE and its localization.

Third, we propose to enhance the INT header with specific fields (e.g. a flag) enabling the UE to request specific network service operations, properly pre-programmed by the SDN Controller/Orchestrator. For example, the UE, according to the experienced e2e latency synchronized with geolocation information, may request traffic steering between cloud and edge resources or migration between nearby edge nodes according to mobility and experienced delay conditions.

Overall, such extensions enable the UE to accurately monitor the experienced performance of the selected 5G service and to take pre-enabled decentralized actions without involving the SDN Controller/Orchestrator when critical conditions appear.

The proposed beyond 5G approach is however not free from limitations. For example, as assessed in this paper, the proposed INT extended to the UE consumes bandwidth resources. In addition, the decentralized UE capability to self optimize its service experience may generate instabilities or inefficiencies in the network if not properly controlled. For example, multiple UEs may simultaneously request migration to the same edge node, possibly exhausting networking and/or computing resources. For this reason, besides leaving the SDN Controller/Orchestrator the possibility to inhibit/pre-empt UE requests of service adaptations, in this work we also present a suitable ML-solution enabling the prediction of UE behaviours thus anticipating service adaptations before QoS degradation occurs. For space reasons, it is left for future works the management and prediction of the behaviour of multiple served UEs, avoiding and controlling the attempts to simultaneously enforce self-adaptions potentially leading to networking and computing resource exhaustion.

The proposed INT extensions and INT-triggered source-based steering are implemented using P4-based INT programmability and are validated in a comprehensive multi-segment testbed including a real cellular domain, a packet-switched backhaul and an SDN disaggregated metro-core optical domain running telemetry against soft failure events. In addition, ML algorithms are also adopted enabling the enforcement of adaptation configurations according to predicted performance. Finally, we discuss open issues and the applicability of the proposed solutions in future real deployment scenarios, with particular focus on scalability, confidentiality and security aspects.

This work is the invited extension of the three-pages paper in [27]. This work introduces specific novelties not included in [27], such as all aspects related to localization and prediction, including the ML solution described in Sec. 3 and the applicability aspects described in Sec. 5.

2. INT IN 5G CONVERGED PACKET-OPTICAL NETWORK: CURRENT LIMITATIONS

Beyond 5G envisions the need of monitoring the end-to-end performance of selected applications offered to end users in the whole communication segment, i.e., from the user equipment to the edge/cloud resource hosting the application, including all the crossed networking segments and thus realizing the so-called cloud continuum awareness.

Referring to the latency monitoring, current solutions are tailored and confined to specific network segments. For example, latency may be monitored at Next Generation eNB (gNB) of the Radio Access Network (RAN), including the wireless link and the 5G fronthaul [28], but with no reference to the actual latency experienced by specific application traffic. In the wired network segment (i.e., backhaul), application traffic may be monitored efficiently resorting to In-Band Telemetry (INT). INT is typically enforced at each SDN programmable switch by pushing a custom header to the considered application traffic flows. The custom header conveys metadata information related to the crossed switch, among which the latency spent by application traffic packets within the switch, i.e., the *hop latency*. The procedure is triggered at the first switch of the domain (i.e., *source switch*), that typically adds a INT global shim header, and is enforced at each switch of the backhaul domain (i.e., *transit switches*) by adding a local metadata header, thus collecting all the crossed intra-switch latency values. The last domain switch (i.e., *sink switch*) is responsible of including its own local header and provide end-to-end transparency to the application by removing the whole extra header. Moreover, the sink exports the whole INT information using an out-of-band channel to the control/monitoring plane using a dedicated INT Report packet, including the latency details collected at each switch.

Current INT implementations are able to provide detailed intra-switch latency information. Such infos are sufficient to compute end-to-end latency only in specific network scenarios. For example, in a single-layer packet-switched wired network, latency variations typically occur inside the switch due to packet processing, packet queuing, congestion, contention and buffering. However, end-to-end latency is subject to significant variations in complex scenarios (i.e., multi-segment, multi-layer, hybrid wired-wireless) that include a variation of link latency. Link latency computations are not computed by INT implementations and are not easy to compute due to the need of synchronization mechanisms among the involved nodes, generally not available in end-to-end scenarios with segmented

domains.

Examples of significant link latency variations, not monitored by INT, include optical multi-layer scenarios and 5G RAN frontahul segments. In the former, an optical connection (i.e., lightpath) established in the optical layer serving a packet-switched link may be subject to recovery or reoptimization procedures (e.g., soft failures detected by optical telemetry trigger the exclusion of the shortest optical link [3]) leading to rerouting over longer distance paths, thus increasing the link latency without affecting the packet-switched layer end points. In the latter, the segment including the combination of wireless link and the different functional split among 5G distributed (DU) and Centralized (CU) units is subject to statistical latency variations due to mobility (e.g., wireless multipath channel), variable number of users cell subscription, queuing delay, transmission processing and frame alignment [29].

A. End-to-end latency monitoring issues: cloud-to-edge scenario

The segmented network scenario considered in this paper is shown in Fig. 1. Mobile User Equipments (UE) are attached to the 5G Network and connected to a cloud data center by means of backhaul and metro-core network segments. Moreover, a number of edge nodes (i.e., E1 and E2) are located at the backhaul forefront to offer low latency services. The backhaul network is made of programmable bare metal switches N1-N6 running INT. Service applications are placed on the cloud and/or the edge nodes, based on the bounded latency service requirements. Typically, they may be deployed in both places. However, due to the limited processing/storage capabilities of the edge, edge instances are run or activated on-demand only when low latency is not guaranteed by the cloud. Following the figure example, the UE is running the applications placed in the cloud experiencing an excessive end-to-end latency $l_c = l_w + l_n$, where l_w is the RAN/ fronthaul contribution (up to N1) and l_n is the overall wired network contribution, including the optical metro-core network. The INT system monitoring is limited to N1-N6 hop latencies, not detecting any issue. The metro-core optical network latency l_o is subject to variations due to optical lightpath recovery onto a longer path, affecting link latency N5-N6. Moreover, the RAN introduces l_w, not monitorable from the application point of view. Possible countermeasures may be taken to fulfill latency requirements. Handover mechanism could steer the service along path N2-N6, however not reducing the latency. Alternative strategies may steer the traffic to local edge node E1(assuming local application pre-deployment), reducing the latency to acceptable values $l_e < l_c$. However, INT is not aware of the l_o and l_w contributions and variations, therefore the UE end-to-end delay can not be monitored properly and the aforementioned edge steering procedures can not be triggered automatically.

B. Latency and location awareness: edge-to-edge scenario

A second scenario is envisioned to discuss the specific information required by the application traffic to automatically ensure quality of service requirements. Fig. 2 shows a scenario involving the UE connected to the 5G RAN and attached to two possible edge nodes E1 and E2. Stringent low latency requirements, combined with advanced network policies, require the UE to be connected to the edge nodes offering the best performance in terms of both offered end-to-end latency and geographical shortest path. For example, specific and critical applications such as autonomous driving services may rely on both latency

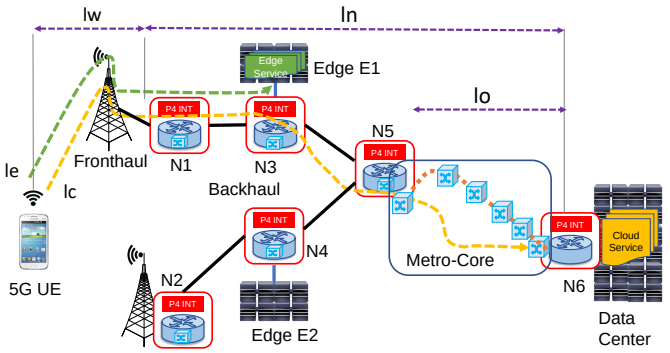

Fig. 1. Scenario of 5G network served by backhaul and metro-core network with cloud and edge resources.

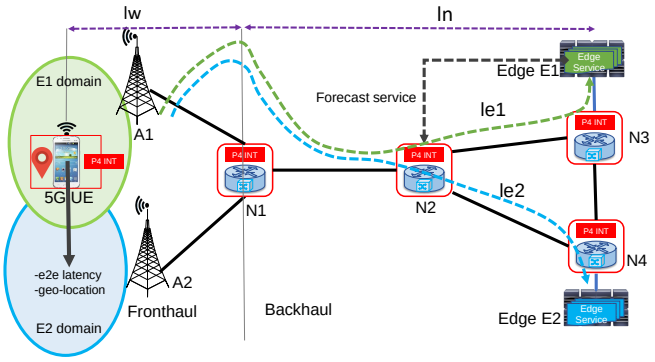

Fig. 2. Scenario with edge-to-edge service steering

and proximity requirements, where the service is offered using a set of edge nodes suitably placed to cover a given geographic transport map. Typically, geographical location is available at the UE thanks to portable positioning sensors (e.g., Global Positioning System - GPS). Geolocation information are retrievable also by using wireless network channel statistics, however they generally lack of accuracy and synchronization updates. Moreover, such information is confined at the 5G system and may not be applied to the specific requirements imposed by the application. In the scenario example of Fig. 2, the UE is attached to antenna A1 and served by edge node E1, experiencing latency l_{e1}. The UE is also moving from the A1 coverage to the A2 coverage. Thus, the 5G network may trigger the handover to A2, allowing the application to follow the A2-N1-N2-N3 path to E1. However, a closer edge E2 would be available serving the UE in the new geographical zone with a reduced end-to-end latency, independently from the RAN handover procedures (i.e., independently from the A1 or A2 handover mechanisms). Unfortunately, end-to-end monitoring is unavailable and the application is unaware of the best edge selection. The availability of detailed geolocation information at the application layer, combined with an efficient forecast service, able to predict the future latency and geolocation, might provide an automatic network-triggered mechanism to steer the traffic towards the most suitable (e.g., closer) edge node.

3. PROPOSED EXTENDED INT AND DECENTRALIZED STEERING STRATEGIES

To enable telemetry-driven in-network steering without the intervention of the SDN controller, we propose an overall extension

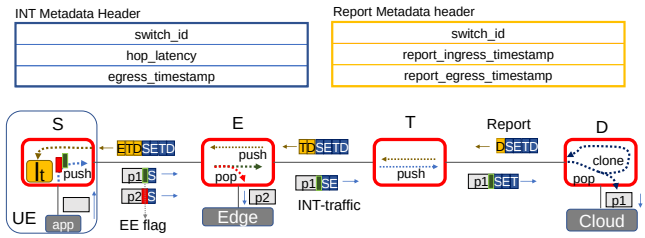

Fig. 3. Extended INT metadata and UE-suggested steering.

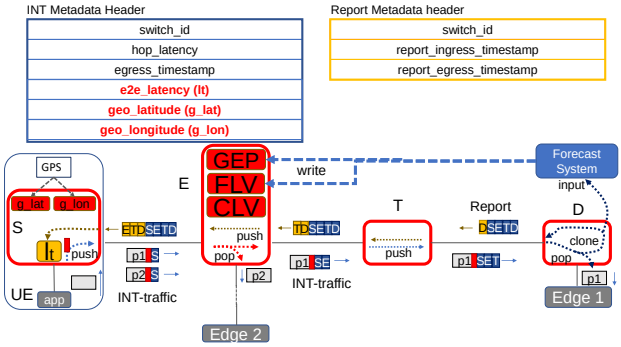

Fig. 4. Detailed INT metadata and forecast-based steering at the P4 switch (FSS).

of the INT architecture that includes the following novelties:

1. The inclusion of the UE in the INT domain;
2. the extension of the INT mechanism providing backward INT information with actual end-to-end latency and geolocation metadata;
3. automatic decentralized steering strategies triggered by extended INT awareness;
4. a forecast AI/ML-based system predicting application specific end-to-end performance and notifying the edge switch for automatic steering.

In the following subsections the first three novelties will be explained. A dedicated section will be devoted to the description of the AI/ML-based forecast system based on INT information.

A. Extended INT and Report

The first innovation relates to the INT domain enlargement, involving the UE. The inclusion of the UE is performed by implementing an SDN P4 programmable switch inside the UE acting as a service app. A possible real implementation may rely on a lightweight virtual container programmed to process only data generated and received by the considered application. In this paper we will refer to upload service traffic. The switch, co-located at the UE, is programmed to act as telemetry source node (i.e., responsible for pushing the shim header and include the UE metadata). With reference to the two considered scenarios (i.e., edge-cloud of Fig. 1 and edge-edge of Fig. 2), the extended INT details are shown in Fig. 3 and Fig. 4, respectively. In both solutions, the INT domain starts at the UE acting as source node (node S in the figures) and terminates at sink node D, co-located with the data center or the edge node, responsible for removing all the INT headers and route the traffic transparently to the application server. Along the end-to-end path, transit nodes T simply add their own INT headers. Among these nodes, special steering switches E are extended to support steering functions to the attached local edge node and, in the case of local edge steering, similarly as node D, are programmed to pop INT headers to provide traffic transparently to the local edge server.

The second innovation, allowing the end-to-end latency computation, envisions an extended INT Report packet processing. Typically, the Report packet is generated as out-of-band packet destined to monitoring plane analysis, reporting the hop latency and egress timestamp information retrieved along the standard INT path forward direction (i.e., S, E, T, D blue headers of Fig. 3). First, node D is programmed to generate and recirculate a Report packet in the backward direction from destination up to the source (i.e., the UE). Second, all the switches are programmed to push the ingress and the egress timestamps related to the Report transit in the backward direction up to S. Both forward and backward timestamps stored in the Report packet (i.e. blue and yellow headers referred to S, E, T, D nodes in Fig. 3) allow all the switches to compute the link latency experienced by the packet. In fact, P4 switches do not use absolute timestamps, but each switch has its reference timestamp, related to the P4 switch bootstrap time. Thus, , the proposed method computes the latency without the need of having all the nodes synchronized, since all the latency contributions are computed as differences between timestamps collected at the same switch.

As an example, when the Report message reaches node S, that P4 switch is able to compute the different contributions by parsing the data in the different INT headers of the Report. By recording its ingress timestamps, S compares it with the egress timestamp related to the forward INT header, recorded by node S. This way S is able to compute the end-to-end latency l_t and store it inside a P4 register. Considering the processing time at each traversed node, node S is able to compute it as the difference among the egress timestamp and the ingress timestamp read from each INT Header (i.e., processing a single INT Header). For the evaluation of the traversed links latency, the solution correlates the information of both the forward and backward INT Headers, registered by the same node (i.e., processing the two INT Headers created by the same node). In this case, S has full end-to-end visibility and may compute and store all the link latencies and the end-to-end latency.

With respect to traditional latency measurements, which use active probes with additional traffic streams generated via application tools (e.g., ICMP, Bidirectional Forwarding Detection, iperf), the proposed solution considers INT as an example of passive probe (i.e., without any additional stream injected in the system). In fact it uses only the actual matched application traffic as probing stream, at the single packet granularity and at the microsecond resolution time.. In addition, being INT transparent up to the transport layer (e.g., Ethernet, IP, MPLS segments), the extended INT is suitable also to measure packet-switched network segments not supporting INT (e.g., layer-3 legacy router domains), provided that the segment boundary nodes employ P4 INT-enabled switches.

In the edge-to-edge scenario, following Fig. 4, besides local timestamps and hop latencies, the UE adds to packet $p2$ its INT header pushing its current position (i.e., the latitude and longitude pair, *geo_latitude* and *geo_longitude*) and the l_t latency experienced by the previous packet $p1$, stored in the local UE P4 switch instance. The current space position information are retrieved from internal UE sensors (e.g., GPS) and stored inside

two dedicated registers (g_lat and g_lon). This allows to store in the same traffic packet the combined synchronized information of latency and geolocation, that is utilized for decentralized steering decisions.

B. Decentralized steering strategies

The third innovation, driven by the extended INT mechanism, resides in the awareness acquired at the different INT nodes, that may be utilized to take forwarding/steering decisions based on the INT information. Different decision strategies are hereafter proposed: the former based on the cloud-edge scenario of Fig. 1, the latter utilizing forecast systems combined with INT information in the edge-edge scenario of Fig. 2. Both steering strategies are decentralized, i.e., not handled and triggered by a central controller.

B.1. UE-suggested steering (UES)

The first decentralized strategy is called *UE-suggested steering* (UES) and include the UE and the switch co-located with the edge node. With reference to the cloud-edge scenario and the INT procedure of Fig. 3, the cloud-edge steering selection is enforced at node E. However, the information source triggering the steering selection is originated by the UE (node S). In fact, the end-to-end latency l_t is computed and stored in S in a dedicated P4 register. Application traffic is generated by the UE and INT-tagged by S with a novel INT shim header flag, named Enable Edge (EE). The flag reports the state of the experienced latency l_t, compared with a pre-defined target bound latency threshold l_{TH}. The flag is initially set to zero (packet p1 with green EE flag) triggering E to steer traffic towards the cloud, and is turned to 1 when $l_t > l_{TH}$. The EE flag is inspected only by switch E. In the case EE is set to 1 (packet p2 with red EE flag), the node automatically steers traffic to the local edge node while popping INT headers. In order to preserve network stability and avoid unexpected steering events, the UE S node is programmed to set the EE flag when a certain threshold number of packets experience $l_t > l_{TH}$. This is realized in the switch by resorting to a dedicated P4 counter.

B.2. Forecast switch-triggered steering (FSS)

The second decentralized strategy is called *forecast switch-triggered steering* (FSS). The utilization of a forecast service at the edge node, processing online synchronized metadata referred to the considered application traffic, is able to predict events to be notified to and enforced at the network layer, enabling automatic steering at the programmable switch level. Fig. 2 and Fig. 4 show the considered solution scenario and the detailed workflow, respectively. In Fig. 2 the mobile UE is connected to edge node E1 generating a traffic stream path N1-N2-N3 having e2e latency $l_{e1} = l_w + l_n$, where l_n is the latency introduce by the wired network (backhaul) and l_w is the latency introduced by the fronthaul including the wireless contribution from UE to the antenna. The S switch adds its current geolocation info and the e2e latency computed resorting to the Report packet mechanism (related to previous application packet p1). The detailed per-packet UE metadata are processed at switch N2 by inspecting INT and are provided to the forecast service located at E1. Combined latency and position thresholds are configured on N2 to trigger traffic steering to edge node E2, switching the stream to path N1-N2-N4 having latency l_{e2}.

Switch N2 takes steering decisions based on combined current and forecast INT processing. The detailed INT-triggered mechanism is shown in Fig. 4, in which N2 is represented by switch E connected to Edge 2 node. Referring to e2e latency, the switch implements a dedicated register that is updated by the forecast service, i.e., Forecast Latency Violation (FLV). In particular, given a pre-determined t_f time window, the forecast system computes and writes the number of future violations that are estimated to occur in the next t_f time interval. The switch performs a similar processing on the actual received data, evaluating the $l_t > l_{TH}$ violation events and storing detected violations in Current Latency Violation (CLV) register. The decision is computed considering the total number of violations (i.e., current and forecast, FLV+CLV), thus anticipating the steering process before the major latency variation events occur. Referring to geolocation, in this case a single register Geo-based Edge Pointer (GEP) is utilized to store the address of the assigned edge node. If forecast system is not utilized, the switch computes the UE-edge distance based on INT information and detects whether the UE is currently positioned within the edge node geographical domain. Alternatively, if the forecast system is employed, it is able to predict the user position in advance, thus notifying to the switch the closest edge. This way, a combined decision considering both latency and location is possible. For example, referring to Fig. 2, the switch, based on the edge address, may perform traffic steering on the port to the newly assigned edge, e.g., switching traffic from N2-N3 (E1 node) to N2-N4 link (E2 node). The system may work also in the cloud-edge scenario, for example preventing a edge steering event in the case geolocation information report an excessive distance or a predicted UE trajectory moving towards the zone domain of another edge node. It is worthwhile to note that the classic handover mechanism, e.g. switching the traffic in the 5G network from antenna A1 to A2, continues to work autonomously and is handled by the 5G network.

C. Forecast System Description

The considered forecast system (FS) receives from the switch responsible of traffic steering the time series of the computed delay value for the specific application and the related user position. Such data are used for both training the AI/ML-based forecasting algorithm and for forecast values inference. Note that, different type of AI/ML techniques can be used and training can be done either in the edge node or in the cloud if more computational resources are needed as reported in [30].

The forecast algorithm considered in this study is based on Long Short-Term Memory (LSTM). LSTM is a special form of Recurrent Neural Network (RNN) that can learn long-term dependencies based on the information gathered in previous steps of the learning process. LSTM consists of a set of recurrent blocks (i.e., memory blocks) where each block contains one or more memory cells and multiplicative units such as *input*, *output* and *forget gate*.

LSTM is one of the most successful model for forecasting long-term time series. The LSTM can be characterized by different hyper-parameters, specifically the number of hidden layers, the number of neurons, and the batch size. Details of LSTM parameters and their impact on prediction accuracy can be found in [31]. However, the process of finding optimal hyper-parameters which minimize the forecasting error could be time and resource consuming.

When LSTM is utilised for forecasting a time series, in general, the input vector/layer corresponds to the n previous data points and the output vector/layer corresponds to k steps ahead with respect to the current time t of the considered time series. In this work, a *stacked LSTM model* is exploited with a multi-step (i.e.,

$k > 1$) forecasting.

In *LSTM multi-step forecasting (LSTM-MSF)*, LSTM predicts k number of data points by considering n previous observed data points.

$$P(t+k, t+k-1, ..., t+1) = model(O(t), O(t-1), \\ ... \quad , O(t-n-1)), \quad \textbf{(1)}$$

where k > 1, P is the prediction of the single data point at time t and O is the observed value at time t.

Note that the offline training is considered in the evaluation, where weights are updated by using the Backpropagation through time (BPTT) [32] gradient-based technique for training the data set.

4. EXPERIMENTAL DEMONSTRATION

The proposed extended INT, combined with the described FS, has been implemented and evaluated in a multi-segment 5G network testbed including cloud and edge platforms. The testbed includes a cellular network segment composed by a RAN fronthaul, and a wired segment including a backhaul P4 programmable switch network and a disaggregated metro-core optical network. The detailed setup of the testbed in shown in Fig. 5.

The RAN segment includes UE, Distributed Unit (DU), Centralized Unit (CU) and the Evolved Packet Core (EPC) components (note that in the experimental setup a 4.5G deployment with an EPC instead of a Next Generation Core – NGC – is considered without impacting the achieved results). The RAN is implemented utilizing the OpenAirInterface (OAI) open source suite. The UE, DU and CU components run in NUC mini-PC, the EPC is deployed in UPBoard mini-PC, while the radio units utilize the Ettus B210 Software Defined Universal Software Radio Peripheral (USRP) devices. The DU-CU interface is configured to run Option 2 functional split using 25 resource blocks and single-input-single-output preference. The UE is deployed with the Behavioral Mode version 2 (BMV2) P4 switch docker container, configured to receive traffic generated by a local Spirent N4U traffic generator and re-transmit it after enforcing source node S INT operation. To guarantee the experimental reproducibility, instead of considering a specific time-sensitive application, constant bit rate traffic (UDP port 5001) is generated by the N4U, emulating the traffic originated by the UE.

The wired backhaul segment comprises three P4 switches configured with the aforementioned extended INT functionalities. The switches are implemented inside bare metal servers (Intel Xeon E5-2643 v3 6-core 3.40GHz clock, 32 GB RAM) using BMv2 and are connected by means of 40Gb/s Ethernet interfaces using Mellanox ConnectX3 Network Interface Cards. The switches run the baseline INT P4 code version 2.1 [33] extended to support e2e latency and geolocation.

The optical segment is composed by a partially disaggregated network made of two Reconfigurable Add-Drop Multiplexers (ROADM) made of two Finisar Waveshaper Wavelength Selective witches (WSS) each, configured as two configurable ROADM degrees, and two attached xPonders (Ericsson SPO coherent 100G cards). ROADMs and xPonder are handled by SDN agents extended with optical telemetry and controlled by the ONOS Controller through NETCONF/YANG API and utilizing the OpenConfig YANG model [34]. Optical telemetry is able to monitor the xPonder receiver Bit Error Rate (BER) and Optical Signal to Noise Ratio (OSNR). Primary O1-O2 and backup

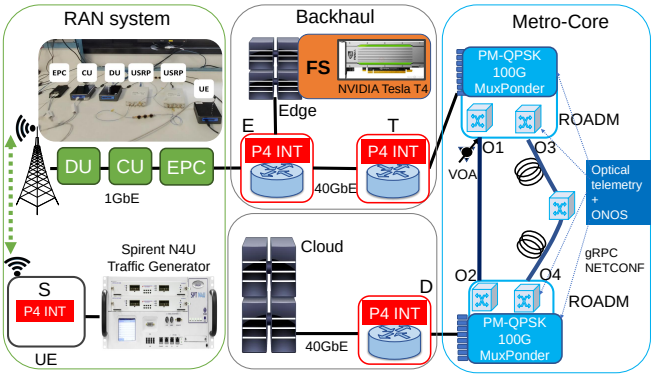

Fig. 5. 5G Packet-Optical network testbed including cloud, edge, UE, P4 switches and FS platform.

O3-O4 optical paths are configurable, the former 1km long, the latter a geographical multi-span link.

The FS, described in Sec. 4. C, is placed in a Docker Container in the edge. It implements the LSTM by using Google's TensorFlow library, accessed through the Keras high-level front-end. The FS exploits an NVIDIA Tesla T4 card as shown in Fig. 5 for both training and testing.

A. UES evaluation

The UES decentralized steering of Sec. 3. B.1 is evaluated by considering the INT code implementing the workflow of Fig. 3. INT functionalities are enforced at nodes S, E, T, D. Initially, upload traffic generated by the UE is destined to the remote cloud server. The considered end-to-end latency threshold l_{TH} is set to 50ms. Different latency conditions are induced in the whole network to trigger UES. In particular, three latency variation steps are enforced. In step 1 (low latency step), T-D multi-layer optical link is configured along O1-O2 path. Optical telemetry outputs BER=$1.9x10^{-8}$ and OSNR=38.5dB. Moreover, in step 1, RAN serves only the UE, thus secondary services are not activated for additional users. In step 2 (medium latency step), the RAN fronthaul is loaded with additional secondary services, inducing an average latency increase. Secondary services are here emulated by running additional 600Mb/s iperf traffic in the DU-CU interfaces. In step 3 (high latency step), in addition to step 2 latency introduced by RAN, a constant latency variation is added due to the optical segment. This is realized through a soft failure recovery. Soft failure is performed manually varying a Variable Optical Attenuator (VOA) placed at the WSS degree optical output line O1. The soft failure causes a signal quality degradation reaching $BER = 10^{-2}$. The telemetry system notifies ONOS, that automatically triggers optical connection rerouting over the longer O3-O4 backup path. This originates a latency step in the optical domain, thus further increasing end-to-end latency.

The plot of Fig. 6 shows the BMv2 switches hop latency values monitored by INT (i.e., intra-switch latencies) using UES. Results show latency distributions limited within 1ms. This means that the added INT load does not impact significantly the end-to-end latency. Moreover, the figure shows that S and D experience higher hop delays. This is an expected result, since the most processing intensive effort (shim header push at S, shim header pop and recirculation at D) is performed by the INT end points. This result suggest that INT is scalable for an increasing number of transit nodes (i.e., considering a bigger topology and longer paths).

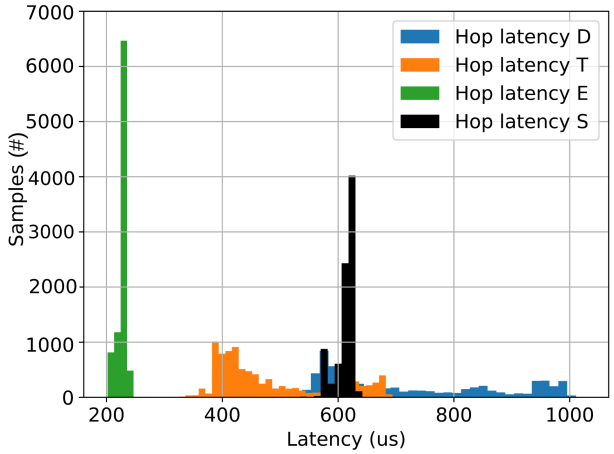

Fig. 6. UES: hop latency at each INT-enabled switch.

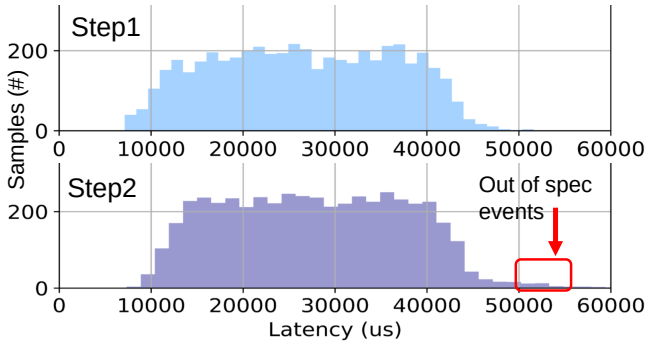

Fig. 7. UES: RAN latency distribution in the overall 5G fronthaul, including wireless connection (link S-E).

RAN latency distribution (S-E link latency) is measured by extended INT computations and depicted in Fig. 7 for step 1 (no secondary services) and step 2-3 (secondary services up). The figure shows that in step 1 the distribution does not exceed the 50ms threshold significantly, while in step 2 a number of sporadic out-of-spec events are registered, however not triggering UES.

UES is actually triggered at step 3 and the steering event is shown in Fig. 8. The plot shows the monitored end-to-end latency stored at S for a traffic flow sequence of 1000 packets per second. The figure also shows the l_{TH} threshold and the number of violation occurrences detected at S. The P4 code in S triggers the EE flag activation when the monitored latency exceed 10 violations. The results show that around 500ms are required to trigger UES after the step 3 latency increase event. The steering is applied immediately upon the reception of the first EE-enabled packet and traffic is redirected to the edge node, thus immediately reducing the perceived average end-to-end latency of around 6ms. Such value corresponds to the latency introduced by the E-D backhaul network segment including the optical backup path. This way, flow end-to-end latency values are restored to in-spec range (i.e., below 50ms).

The details of INT headers for UES are shown in Fig. 9. The figure reports a Wireshark capture performed at S (i.e., the UE, source IP address 10.0.0.48). The INT header is pushed after the UDP header. The INT shim header includes the EE flag (set to

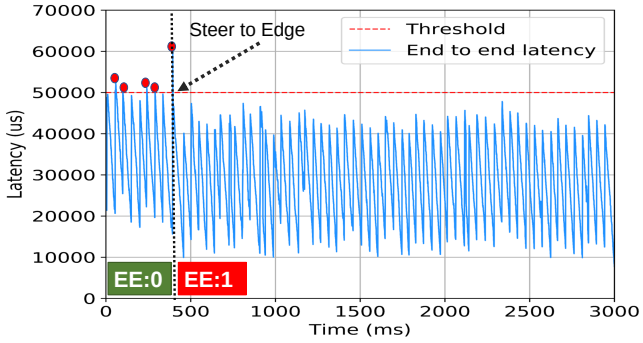

Fig. 8. UES: Monitoring of the e2e latency in step 3 and cloud-edge steering event.

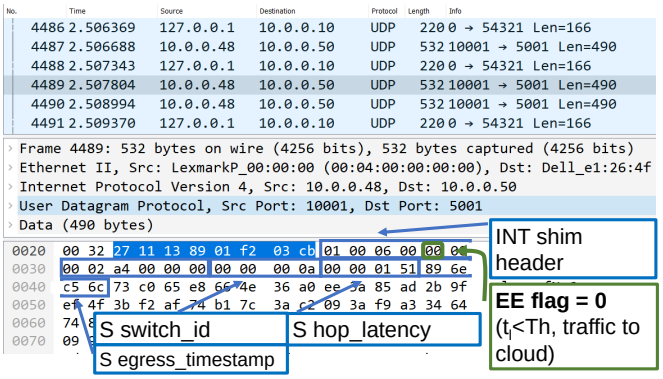

Fig. 9. UES: Wireshark capture of INT-tagged traffic at the UE.

0, captured at step 1 when the flow is steered to the cloud), the switch id (0x10, corresponding to node S), the hop latency (0x151, corresponding to 337us) and the egress timestamp (0x896ec56c, corresponding to around 38 minutes after the P4 switch bootstrap). With respect to standard INT, the extensions overhead is 0 bytes for forward INT and 12 bytes per node for the Report in backward INT.

B. FSS evaluation and AI/ML-based forecast results

The FSS evaluation is run over the same testbed and experimental conditions with respect to UES, however in this case the combined awareness of end-to-end latency and geolocation are exploited by the FS to predict future latency values and position to allow the P4 switch to anticipate network layer decisions (e.g., traffic steering) and reduce the time for which performance requirements are not satisfied. A python script updates the UE latitude position emulating a starting position at 100m from the edge node with 200m/s uniform speed mobility towards the edge domain border (i.e., the UE is quickly moving outside the edge domain). The domain border is set at 200m from the edge. This constraint is mapped in the P4 code, so that the local steering option is active only if the UE-edge distance, computed using the current UE and the edge node coordinates, does not exceed 200m.

The details of INT headers for FSS are shown in Fig. 10, reporting the Wireshark capture at S. With respect to UES, the INT header is enriched with the geolocation information pushed by the UE. Geolocation info (i.e., latitude and longitude) are retrieved from the UE GPS with 2 s sampling time, written in the P4 registers (i.e., g_lat and g_lon) using the BMv2 runtime CLI and are encoded with 4-byte long GPS decimal representation.

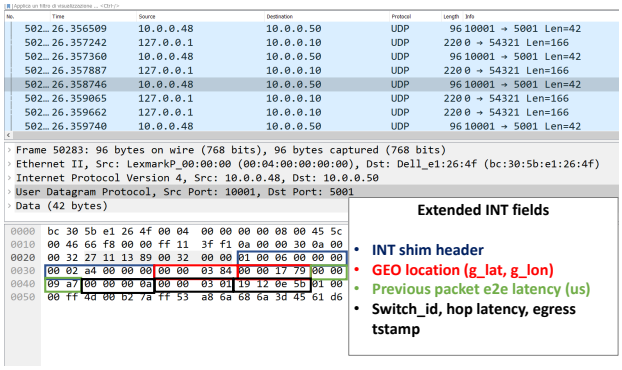

Fig. 10. FSS: Wireshark capture of INT-tagged traffic at the UE.

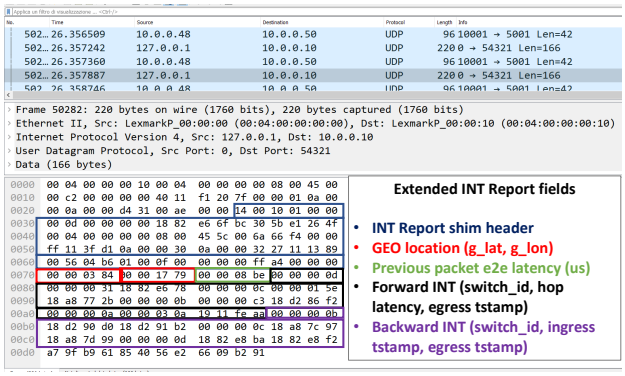

Fig. 11. FSS: Wireshark capture of the extended INT Report traffic at the UE.

Fig. 11 shows the details of the Report packet received by S at the end of the forwards and backward INT. In particular, the backward INT headers collected along the path and received by S are visible (node_id values 0x0d, 0x0c and 0x0b identify nodes D, T, E, respectively, along with their ingress and egress timestamps). By adding its own values, S will compute the packet latency and store the value in the next application packet INT header. With respect to standard INT, the FSS extensions overhead is 12 bytes for forward INT and 12 bytes per node for the Report in backward INT.

Fig. 12 shows the hop latencies introduced by each P4 switch of the path, including S, in the INT forward direction. With respect to the UES results, we notice a slightly larger latency distribution at node S, ranging from 200us up to 1ms. This is due to the extra processing required at S that includes the geolocation update process and the extended INT processing including the novel location and latency fields. However the observed latency is always below acceptable values (i.e., 1 ms). These results confirm the feasibility of the proposed extended INT, obtained with the BMv2 software switch, not designed to achieve high performance results. Implementations over P4-capable programmable hardware (e.g., Tofino switches) are expected to dramatically decrease such processing delay values.

The FS is placed at the edge node and it receives INT packets by configuring E switch to clone the selected packets and direct them locally to the FS IP address, without removing INT.

As described in the above subsection, the data sets are obtained experimentally by generating 1000 INT packets per second with $1ms$ intervals in three different latency step scenarios such as *low latency*, *medium latency* and *high latency*. The input

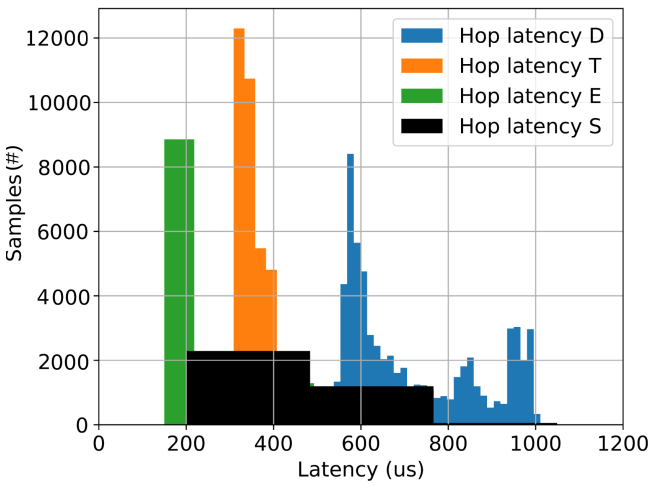

Fig. 12. FSS: INT-monitored hop latency at each switch.

features to the LSTM algorithm are the latency and the position (i.e., latitude and longitude) up to n INT packets back from the current time t. The LSTM output are the latency and the position at each of the next k INT packets from the current time t.

After performing each prediction, the FS triggers E registers write actions by resorting to the BMv2 remote runtime CLI using a Thrift-based API. In particular, it dynamically updates the GEP and the FLV registers.

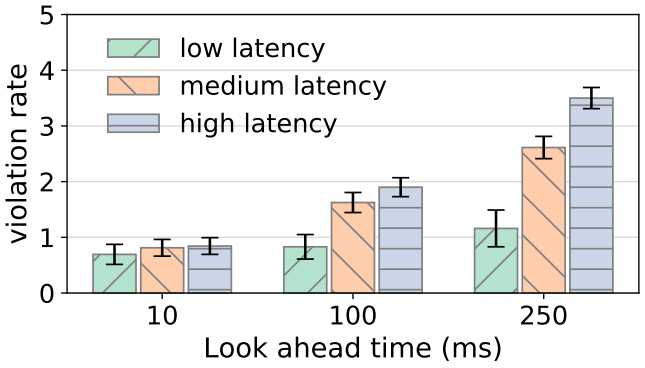

Fig. 13. Violation rate as function of look ahead time.

The performance evaluation of the LSTM utilized by the FS is conducted by considering a training versus testing proportion of 60 : 40, implying that 60% of the data is used for training and 40% for testing. This split ratio was selected as it was able to provide the lowest Root Mean Square Error (RMSE) compared with 70 : 30 and 80 : 20 data set split ratio. The input feature size is equal to $n = 50$ latency and position values corresponding to a look back time t_b of 50ms. Three different look ahead times are considered: 10ms, 100ms, and 250ms. In the considered LSTM, the number of hidden layers is set to 2 and the neurons in the hidden layer is set to 100. The epochs and the batch size are set to 50 and 10, respectively.

Tab. 1 shows the computation time needed by the FS for the training and testing phases during all three look ahead times. As shown, the performance in terms of computation time is similar in all the three cases for both the testing and training phase. In

particular, it must be highlighted that prediction time is about $21ms$.

Table 1. LSTM: Training and Prediction Time

Scenarios	Look back time = 50ms					
	Look ahead time [10ms]		Look ahead time [100ms]		Look ahead time [250ms]	
	Training Time [s]	Prediction Time [s]	Training Time [s]	Prediction Time [s]	Training Time [s]	Prediction Time [s]
low latency	338.75	0.02126	336.16	0.02102	329.13	0.02149
medium latency	339.23	0.02134	336.36	0.02149	334.26	0.02118
high latency	339.37	0.02131	337.06	0.02136	341.37	0.02143

During the testing phase, the violation rate is considered as an additional performance parameter. The violations rate is defined as the ratio between the forecast violations and the number of effective violations in the real test data set. Thus, the violation rate provides an estimate of the forecast precision.

Fig. 13 shows the violations rate as a functions of the considered LSTM output size (i.e., the look ahead time t_f computed as $t_f = k * 1ms$). As shown in Fig. 13, the FS underestimates the number of violations if the look ahead time is $10ms$ in all the steps. On the other hand the overestimation of the violation increases as a function of a look ahead time increase for all three steps. Note that, during the testing phase, ten violation estimations are computed and the results are plotted with 95% confidence interval.

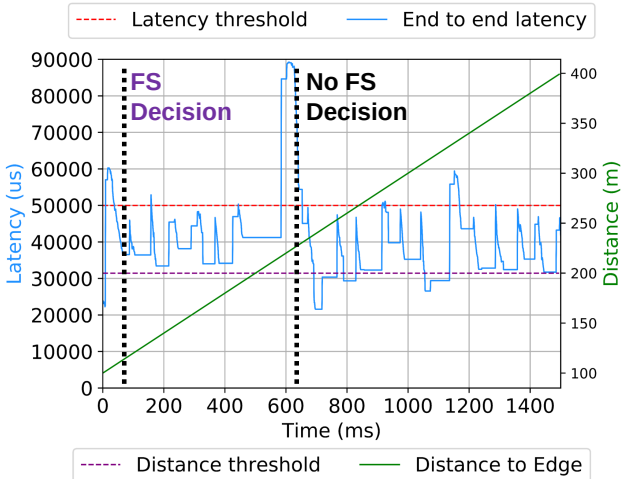

Fig. 14. FSS: Step 3 decentralized steering decisions.

Fig. 14 shows the real experiment monitoring of the end-to-end latency (step 3, high latency) and the geolocation observed by switch E and the different decisions taken by the switch with and without the FS support. The P4 code in E triggers the steering if the monitored latency exceed 79 violations and the monitored geolocation distance does not exceed 200m. The considered 79 violations parameter correspond to the FS predicted number of violations with 50ms look back time and 10ms look ahead time (i.e., when the violation rate is close to 1, thus avoiding FS false alarm predictions), see Fig. 13. With no FS support, the decision is taken after 631 ms, with a delay similar to the UES case, in which only latency was monitored. In particular, switch E, differently from UES case, decides to not steer traffic to the local edge since the current geolocation distance exceeds the 200 m distance threshold (i.e., the UE is outside the edge domain). This behavior suggests that the combined use of latency and geolocation is beneficial, since it avoids unwanted, multiple or out-of-dated steering decisions. We repeated the experiment using the FS and in this case the decision is assumed in advance. In this case, the steering decision is enforced on the forecast results. The FS estimates the latency violation threshold in advance (resorting to the 10ms look ahead time estimation), however, at the same time, forecasts a rapid UE movement outside the edge domain boundary. The result is that FLV is updated in advance, while GEP is not updated with the local edge address, thus denying the steering. The overall decision delay includes the 50ms FS look back time, the FS prediction time (21ms) and the thrift-based notification changing the FLV registers at switch E (1ms). The overall decision (no steer) is performed after 72 ms. This demonstrates the FS-driven steering decision time improvement of around one order of magnitude. Moreover, the decision is assumed when the UE is moving within the edge domain, showing that the forecast system, by predicting the future out-of-border trajectory behavior, has the benefit effect to increase the overall network awareness, potentially enabling early alert notifications to the 5G/SDN controller to react in advance with either handover or alternative decentralized steering decisions at different nodes (i.e., pre-plan the steering in the nodes of the next edge domain in advance for the UE entering in the next edge domain, with no traffic disruption or high latency traffic bursts).

5. INT AT THE UE APPLICABILITY

The results shown in Sec. 4 suggest that the proposed INT-based decentralized steering decisions including the UE involvement may improve in-network dynamic operation thanks to situational awareness information such as identity, position, time, device, function. However, a future real deployment will face a number of relevant open issues. Overall, we identify three main aspects: scalability, network leakage and security.

The overall system scalability when a high number of UEs use INT for a number of different time-sensitive applications requires careful design strategies. This is mainly reflected in the scalability evaluation of P4 switches. The literature confirms that the latency performance of P4 switches deployed in hardware (bare metal, FPGA) scale with the number of installed flow entries due to the ternary content-addressable memory (TCAM) hashing mechanism [13]. The proposed implementation employs different P4 codes at the UE (S), the edge switch (E) the transit (T) and the sink (S) nodes deployed in different platforms (i.e., S as a virtual P4 service /container, the other nodes as bare metal switches). Concerning the number of involved UEs, no significant scalability issues arise, given that the geolocation info are sourced by each UE resorting to a fix number of registers (g_lat and g_lon in S) and E node stores its own location information and the GEP pointer. No significant issues are identified for INT nodes T and S, running stateless INT operation. About the per-packet INT monitor, in the case external collectors are unable to process data at high speed, the possibility of performing time-window metadata statistics extractions has been recently demonstrated in P4 [35]. Concerning the number of applications, a possible scalability bottleneck is identified in node E since the violation registers (CLV and FLV) size scales with the product of the involved UEs and the number of applications. In this case, the design of the P4 switch will require a fixed size of

register arrays, accessible through flow entry matching either the UE and application pair, or the INT flow_id, currently under discussion in the P4 INT group. Such design ensures the switch latency scaling via flow entry size. Careful design will need to be studied for the FS, since in this paper the AI engine is trained using the dataset corresponding to the behavior of a single UE running a single application in different network scenarios. A deployment prototype should take care of the latency/geolocation behavior of multiple UEs, possibly served with different scheduling and priority options in the network based on the 5G service delay requirements. Finally, at the UE, the co-located P4 switch service needs to store the latencies and the thresholds of each application, involving a limited amount of registers and memory (i.e., 2x 32 bit register per application). In this case, a careful mobile-oriented implementation design should be able to support INT with a reduced CPU and memory processing effort. For example, INT functions may be offloaded directly inside the UE network interface card.

A second issue is related to the network leakage. In the paper the UE is able to collect detailed information on application latency values related to different segments of the network. This may pose confidentiality issues while spanning different provider domains or when the INT information granularity may allow to infer performance bottlenecks of data center/edge or telco operator. In this case, possible countermeasures in a real deployment may rely on authorizing only end-to-end INT information and measurements at the UE (e.g., end-to-end latency to cloud and edge), while removing the network segment information, that may remain available only for the INT-enabled switches to run decentralized steering.

About security, the UE potentially may generate unwanted network attacks due to malware exploiting the INT service. More in general, this aspect is potentially relevant and needs to be further investigated and detailed. In this paper we modeled the INT service at the UE as a P4 container. In a real deployment involving different UEs (e.g., sensors, vehicles, mobile phones, cameras), enabling the UE to generate and process INT-aware flows should be designed as a dedicated network service inside the UE. Such service will need to be deployed in the operating system (OS) space (when applicable) as a critical service and run in a protected OS memory area. The applications that desire to utilize such service must run preliminary secure channels for authentication and authorization purposes. Moreover, to protect against possible malware, automatic security checks should be run by the OS to verify that the local app is not corrupted and data generated by the app are compliant with the expected traffic flow features (addresses, protocol stacks, L4 ports) used by the application.

Finally, the proposed solutions are not conceived as a step back towards traditional networking resorting to distributed protocols. Conversely, they are introduced as an evolution of the SDN framework, in which a SDN controller is in charge of activating or deactivating application traffic enhanced with UE-INT and decentralized steering. The proposed extensions are possible thanks to SDN data plane programmability. In this regard, the SDN controller is aware of any steering option that is enforced dynamically at the switch level. Such level of dynamicity could not be enforced directly by a central controller in a proper and prompt way, for scalability reasons. In this vision, the SDN controller delegates specific functions to be offloaded at the UE and the switch level.

6. CONCLUSION

In this work, we proposed to extend the use of P4-based in-band telemetry (INT) up to the 5G user equipment (UE) enabling accurate monitoring of the whole end-to-end path including both the wired and wireless segments. Accurate monitoring is achieved by including within INT, besides latency information, also synchronized localization data. Furthermore, we proposed to enhance the INT header with specific fields enabling the UE to autonomously trigger specific network service operations, such as steering from cloud to pre-programmed edge resources. This way, SDN Controller/Orchestrator intervention is avoided during critical operational conditions. Finally, AI-assisted forecasting leveraging on INT latency and geolocation data is also applied to predict such critical conditions and trigger fast edge steering before actual service degradation is experienced. The proposed P4 INT extensions, INT-triggered source-based steering, and INT-driven AI forecasting were validated in a comprehensive multi-segment testbed including a real cellular domain, a packet-switched backhaul and an SDN disaggregated metro-core optical domain running telemetry against soft failure events. Results showed the effectiveness of the system to steer cloud traffic to the edge upon out-of-spec e2e latency without any controller intervention.

FUNDING

ECSEL Joint Undertaking (JU) BRAINE (G.A. 876967).

ACKNOWLEDGMENTS

This work has received funding from the ECSEL Joint Undertaking (JU) BRAINE Project, under grant agreement No 876967. The JU receives support from the European Union's H2020 research and innovation programme and from Italy Ministry of Education, University and Research (MIUR).

REFERENCES

1. P. Schulz, M. Matthe, H. Klessig, M. Simsek, G. Fettweis, J. Ansari, S. A. Ashraf, B. Almeroth, J. Voigt, I. Riedel *et al.*, "Latency critical iot applications in 5g: Perspective on the design of radio interface and network architecture," IEEE Commun. Mag. **55**, 70–78 (2017).
2. Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," IEEE Commun. Surv. Tutorials **19**, 2322–2358 (2017).
3. F. Paolucci, A. Sgambelluri, F. Cugini, and P. Castoldi, "Network telemetry streaming services in SDN-based disaggregated optical networks," J. Light. Technol. **36**, 3142–3149 (2018).
4. Y. Lee, R. Vilalta, R. Casellas, R. Martínez, and R. Muñoz, "Scalable telemetry and network autonomics in actn sdn controller hierarchy," in *2017 19th International Conference on Transparent Optical Networks (ICTON),* (IEEE, 2017), pp. 1–4.
5. K. S. Mayer, J. A. Soares, R. P. Pinto, C. E. Rothenberg, D. S. Arantes, and D. A. Mello, "Soft failure localization using machine learning with sdn-based network-wide telemetry," in *2020 European Conference on Optical Communications (ECOC),* (IEEE, 2020), pp. 1–4.
6. S. Tang, J. Kong, B. Niu, and Z. Zhu, "Programmable multilayer INT: An enabler for AI-assisted network automation," IEEE Commun. Mag. **58**, 26–32 (2020).
7. P. Manzanares-Lopez, J. P. Muñoz-Gea, and J. Malgosa-Sanahuja, "Passive in-band network telemetry systems: The potential of programmable data plane on network-wide telemetry," IEEE Access **9**, 20391–20409 (2021).
8. A. G. Castro, A. F. Lorenzon, F. D. Rossi, R. I. T. Da Costa Filho, F. M. V. Ramos, C. E. Rothenberg, and M. C. Luizelli, "Near-optimal probing

planning for in-band network telemetry," IEEE Commun. Lett. pp. 1–1 (2021).
9. B. Niu, J. Kong, S. Tang, Y. Li, and Z. Zhu, "Visualize your ip-over-optical network in realtime: A p4-based flexible multilayer in-band network telemetry (ml-int) system," IEEE Access **7**, 82413–82423 (2019).
10. I. Pelle, F. Paolucci, B. Sonkoly, and F. Cugini, "Telemetry-driven optical 5G serverless architecture for latency-sensitive edge computing," in *2020 Optical Fiber Communications Conference and Exhibition (OFC),* (2020), pp. 1–3.
11. F. Cugini, P. Gunning, F. Paolucci, P. Castoldi, and A. Lord, "P4 in-band telemetry (INT) for latency-aware VNF in metro networks," in *Optical Fiber Communication Conference (OFC) 2019,* (Optical Society of America, 2019), p. M3Z.6.
12. L. Tan, W. Su, W. Zhang, J. Lv, Z. Zhang, J. Miao, X. Liu, and N. Li, "In-band network telemetry: A survey," Comput. Networks p. 107763 (2020).
13. F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, and P. Castoldi, "P4 edge node enabling stateful traffic engineering and cyber security," J. Opt. Commun. Netw. **11**, A84–A95 (2019).
14. K. Samdanis and T. Taleb, "The road beyond 5G: A vision and insight of the key technologies," IEEE Netw. **34**, 135–141 (2020).
15. B. Zhou, A. Liu, and V. Lau, "Successive localization and beamforming in 5g mmwave mimo communication systems," IEEE Transactions on Signal Process. **67**, 1620–1635 (2019).
16. H. Hantouti, N. Benamar, T. Taleb, and A. Laghrissi, "Traffic steering for service function chaining," IEEE Commun. Surv. Tutorials **21**, 487–507 (2019).
17. M. A. Togou, D. A. Chekired, L. Khoukhi, and G. Muntean, "A hierarchical distributed control plane for path computation scalability in large scale software-defined networks," IEEE Transactions on Netw. Serv. Manag. **16**, 1019–1031 (2019).
18. P. Kokkinos, P. Soumplis, and E. A. Varvarigos, "Pattern-driven resource allocation in optical networks," IEEE Transactions on Netw. Serv. Manag. **16**, 489–504 (2019).
19. A. P. Vela, M. Ruiz, F. Fresi, N. Sambo, F. Cugini, G. Meloni, L. Potì, L. Velasco, and P. Castoldi, "Ber degradation detection and failure identification in elastic optical networks," J. Light. Technol. **35**, 4595–4604 (2017).
20. L. Velasco, L. M. Contreras, G. Ferraris, A. Stavdas, F. Cugini, M. Wiegand, and J. P. Fernandez-Palacios, "A service-oriented hybrid access network and clouds architecture," IEEE Commun. Mag. **53**, 159–165 (2015).
21. D. Rafique and L. Velasco, "Machine learning for network automation: overview, architecture, and applications [invited tutorial]," IEEE/OSA J. Opt. Commun. Netw. **10**, D126–D143 (2018).
22. M. Balanici and S. Pachnicke, "Classification and forecasting of real-time server traffic flows employing long short-term memory for hybrid e/o data center networks," IEEE/OSA J. Opt. Commun. Netw. **13**, 85–93 (2021).
23. Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic flow prediction with big data: A deep learning approach," IEEE Transactions on Intell. Transp. Syst. **16**, 865–873 (2015).
24. W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," IEEE Transactions on Intell. Transp. Syst. **15**, 2191–2201 (2014).
25. Y. Hua, Z. Zhao, R. Li, X. Chen, Z. Liu, and H. Zhang, "Deep learning with long short-term memory for time series prediction," IEEE Commun. Mag. **57**, 114–119 (2019).
26. J. Xie, F. R. Yu, T. Huang, R. Xie, J. Liu, C. Wang, and Y. Liu, "A survey of machine learning techniques applied to software defined networking (sdn): Research issues and challenges," IEEE Commun. Surv. Tutorials **21**, 393–430 (2019).
27. D. Scano, F. Paolucci, K. Kondepu, A. Sgambelluri, L. Valcarenghi, P. Castoldi, and F. Cugini, "Augmented in-band telemetry to the user equipment for beyond 5g converged packet-optical networks," in *2020 European Conference on Optical Communications (ECOC),* (2020), pp. 1–4.
28. G. T. 28.552, "3rd generation partnership project; technical specification group services and system aspects; management and orchestration; 5g performance measurements (release 17)," V17.1.0 (2020).
29. I. Parvez, A. Rahmati, I. Guvenc, A. I. Sarwat, and H. Dai, "A survey on low latency towards 5G: RAN, core network and caching solutions," IEEE Commun. Surv. Tutorials **20**, 3098–3130 (2018).
30. V. Reddy Chintapalli, K. Kondepu, A. Sgambelluri, A. Franklin A, B. Reddy Tamma, P. Castoldi, and L. Valcarenghi, "Orchestrating edge- and cloud-based predictive analytics services," in *2020 European Conference on Networks and Communications (EuCNC),* (2020), pp. 214–218.
31. S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural Comput. **9**, 1735–1780 (1997).
32. Y. Chauvin and D. E. Rumelhart, eds., *Backpropagation: Theory, Architectures, and Applications* (L. Erlbaum Associates Inc., USA, 1995).
33. P. A. W. Group, "In-band network telemetry (int) dataplane specification" version 2.1," https://github.com/p4lang/p4-applications/blob/master/docs/INT_v2_1.pdf (2020).
34. A. Sgambelluri, A. Giorgetti, D. Scano, F. Cugini, and F. Paolucci, "OpenConfig and OpenROADM automation of operational modes in disaggregated optical networks," IEEE Access **8**, 190094–190107 (2020).
35. F. Musumeci, V. Ionata, F. Paolucci, F. Cugini, and M. Tornatore, "Machine-learning-assisted ddos attack detection with p4 language," in *ICC 2020 - 2020 IEEE International Conference on Communications (ICC),* (2020), pp. 1–6.