A Comprehensive Characterization of Threats Targeting Low-Latency Services: The Case of L4S

Marius Letourneau, Guillaume Doyen, Rémi Cogranne, Bertrand Mathieu

A Comprehensive Characterization of Threats Targeting Low-Latency Services: the Case of L4S

Marius Letourneau[1*], Guillaume Doyen[2], Rémi Cogranne[1] and Bertrand Mathieu[3]

[1]LIST3N, Université de Technologie de Troyes, Troyes, France.
[2]OCIF - IRISA (UMR CNRS 6074), IMT Atlantique, Rennes, France.
[3]Orange Innovation, Lannion, France.

*Corresponding author(s). E-mail(s): marius.letourneau@utt.fr;
Contributing authors: guillaume.doyen@imt-atlantique.fr;
remi.cogranne@utt.fr; bertrand2.mathieu@orange.com;

Abstract

New services with low-latency (LL) requirements are one of the major challenges for the envisioned Internet. Many optimizations targeting the latency reduction have been proposed, and among them, jointly re-architecting congestion control and active queue management (AQM) has been particularly considered. In this effort, the L4S (Low Latency, Low Loss and Scalable Throughput) proposal aims at allowing both Classic and LL traffic to cohabit within a single node architecture. Although this architecture sounds promising for latency improvement, it can be exploited by an attacker to perform malicious actions whose purposes are to defeat its LL feature and consequently make their supported applications unusable. In this paper, we exploit different vulnerabilities of L4S which are the root of possible attacks and we show that application-layer protocols such as QUIC can easily be hacked in order to exploit the over-sensitivity of those new services to network variations. By implementing such undesirable flows in a real testbed and characterizing how they impact the proper delivery of LL flows, we demonstrate their reality and give insights for research directions on their detection.

Keywords: Networking, Security, Low-Latency, L4S

1 Introduction

Year after year, network evolutions (e.g., fiber for wired networks, 5G for wireless) enable higher throughput and lower latency delivery, leading to the emergence of new services. The last ones are those belonging to the so-called Low-Latency (LL) applications, such as cloud gaming, cloud robotics and tele-robotics, tactile internet, among others. These applications require the delivery of contents in the order of few milliseconds. As future networks will allow the delivery of such latency constrained applications, they should not penalize other types of services. The Low Latency, Low Loss and Scalable Throughput (L4S) architecture, first proposed in [1], is currently discussed at the IETF [2] and acts as a promising candidate solution to ensure these low network latency requirements.

However, if such a novel architecture exhibits satisfying performance under normal traffic conditions [5], the question of its capability to deal with abnormal traffic is still an open issue. For instance, the ability of L4S to satisfy LL requirements while maintaining a good balance with Classic traffic makes it highly sensitive to non-regular traffic, as illustrated in [4] which studied the impact of traffic burstiness on the L4S forwarding performance. Besides, malicious users could easily exploit such weaknesses to pollute the network traffic and degrade the Quality of Experience (QoE) of consumers of LL applications. This is already the case of cloud gaming attacks which, by leveraging boosters [6], are able to target a set of users playing a common match, so as to provide a poor QoE making the game eventually unplayable.

In a previous work [7] we explored three examples of undesirable flows, which can be considered as attacks that a malicious user can implement against L4S. This is a preliminary work that we extend in this paper, mainly with new evaluations regarding network conditions and a refinement of the potential threats. Our new contributions include:

- A refinement of the experimental conditions we set up to provide more realistic results. As such, those experimental conditions are detailed and evaluated (e.g. impact of instrumentation and performance evaluation of traffic sources);
- An improvement of three relevant threats altering the low latency performance expectation of L4S, by refining our previous work in which some tuning biases have been now resolved;
- The enlargement of the sole three considered cases previously to reach a global coverage of the potential threats (or absence of) by considering the network throughput, the type of queue (i.e. low latency vs. classic) and the type of undesirable flow as main input factors;
- An in-depth analysis of the results leveraging a Principal Component Analysis (PCA) to identify data correlation that would characterize each threat

scenario and pave the way for some future work toward undesirable flows detection;
- The availability of the entire dataset collected during this experimental campaign on the supporting research project website[1] to contribute to a reproducible research.

The rest of the paper is structured as follows. Section 2 provides some background on L4S which lies at the core of our contribution. Section 3 presents a review of past and current work focusing on the different threats targeting network flows with the aim to degrade the network performance and eventually a service availability. Section 4 depicts the testbed we have implemented, the reference traffic we have considered and the leveraged means to implement some undesirable flows emitted by a malicious third party. Section 5 exhaustively presents the results of three references scenarios we selected. Section 6 further explores the conditions under which undesirable flows may actually impact the L4S architecture and provides a correlation analysis of attack data to highlight some first elements regarding undesirable flows detection. Finally, Section 7 concludes the paper and provides some directions regarding future work in this area.

2 Background on the L4S Architecture

The L4S architecture is currently under standardization at the IETF [2] and it focuses on reducing queuing delay for flows with a low-latency requirement. Coexistence and fairness between low-latency flows and Classic (best effort) ones are strong prerequisites in the design of L4S. This feature is achieved by leveraging a scalable congestion control such as Prague [8], Explicit Congestion Notification (ECN) [9] and a Dual Queue Coupled Active Queue Management (DQC AQM) [1, 10].

From the endpoint side, congestion control and the network stack is adapted to improve scalability and RTT-independence. Prague is known as such a scalable congestion control and it is available with TCP [8] or with some QUIC implementations. The main idea is to adapt the reduction of the congestion window to the actual level of congestion instead of drastically reducing the emission rate. This requires an accurate feedback from the network about the congestion level, and this is accomplished with a modified version of ECN called Accurate ECN [3]. ECN lets the network express congestions by marking IP packets when the buffer queue occupation of a network equipment is increasing too much. Then, the endpoint receiving this marked packet must inform the sending endpoint of the connection that a congestion occurred and the latter must decrease its sending rate.

From the network side, a Dual Queue Coupled AQM, composed of three main elements, is proposed. The first component is a packet classifier that differentiates Classic flows and LL ones by checking the ECN flags of the

[1] Upon paper acceptance the dataset collected will be made available at https://www.mosaico-project.org

IP header. A flag from the reserved field of the IP header is used for the ECN codepoint differentiation. The second component is a coupling mechanism which is the core of the L4S proposal; as such, we detail it subsequently. Finally, the last component of the architecture is a scheduler which manages the sending of the packets.

The main component of the L4S architecture is a coupling mechanism (e.g., [1]) which works as follows. To manage the Classic queue, a PI^2 AQM is proposed [11] and is easy to implement [12]. A PI^2 controller generates a packet marking probability in order to be always as close as possible to a target τ_0, which can be an amount of queuing delay or an amount of packets in the queue. This marking probability $p(t)$, a.k.a. the base probability, is governed by the following equation:

$$p(t) = p(t-T) + \alpha \times (\tau(t) - \tau_0) + \beta \times (\tau(t) - \tau(t-T)) \quad (1)$$

where τ_0 is the target value, T is the period used to recompute the probability, set by default to 16 ms, α is a gain factor associated with the error regarding the target and β is a gain factor associated with the variation compared to the precedent computed probability. α is set to 0.16 Hz and β to 3.2 Hz by default, which means that the AQM is more sensitive to variation than to error.

For the LL queue, marking is controlled either with a simple threshold that helps to compute the related mark/drop probability without introducing additional delay or with the base probability $p(t)$ explained previously multiplied by a coupling factor k (by default set to 2, but its choice is left to network operators). To ensure the fair bandwidth sharing between the two types of traffic, the probabilities are coupled before the final decision for marking/dropping. Thus, depending on the Classic queue, LL packets can be marked in order not to penalize Classic flows but never dropped, whereas Classic packets can be marked (for Classic flows supporting ECN) or dropped. This coupling mechanism between both queues is a key concept of L4S, the gist of the idea behind this, is to ensure that low-latency flows do not generate starvation of Classic flows. Without the coupling mechanism, starvation may happen because of the over-marking generated by the router to support the high feedback requirements of scalable congestion controls, which results in an over-reaction from Classic flows which in return reduce their bandwidth more than necessary, leading to a suboptimal networking conditions and performance degradation for all Classic flows. Hence, the coupling factor k acts as a tuning parameter that determines the balance between L4S and Classic flow rates.

3 Related Work

Over the years, the availability of services has always been a concern in networking environments. Legacy volumetric Denial of Service (DoS) attacks [13] have been progressively refined by attackers to lead to a large set of means

[2] Proportional Integral

whose purpose still remains to undermine the service availability to legitimate end users or its proper operation. Low-latency services cannot avoid this and can also be subject to attacks targeting the latency constraints. Particular research and standardization efforts are brought either to intrinsically protect them against such threats or to detect and mitigate them when they occur. For example, Ergenç *et al.* [15] and RFC-9055 [16] propose a comprehensive attack surface identification accompanied by some guidelines for further detection and protection mechanisms focusing on the Time Sensitive Networking (TSN) and Deterministic Networking (DetNet) architectural models [14], respectively. Similarly, the 5G URLLC (Ultra-Reliable Low Latency Communications) group devotes some substantial efforts to ultra-low latency service security [17] since a malicious usage of protocols can lead to a quality reduction or a complete denial of the service delivery [18]. However, despite their relevance to their architectural context, these solutions cannot apply to L4S, which is the architectural model we focus on in the following.

In the subsequent sections, we review the different forms of deviant behaviors in protocol usages which can be deliberately implemented by any malicious third party with the aim to degrade a service availability in the context of the L4S architecture. To that aim, we select those scenarios which apply to several networking environments but exhibit an applicability in L4S. As illustrated in Figure 1, we classify the set of resulting network flows into three sets we detail subsequently, referred to as *misbehaving flows*, *unresponsive flows* and *malformed flows*. Overall, we use the term *undesirable flows* as an abstraction to designate any of them regardless of the underlying purpose as they can be either legitimate or attack flows.

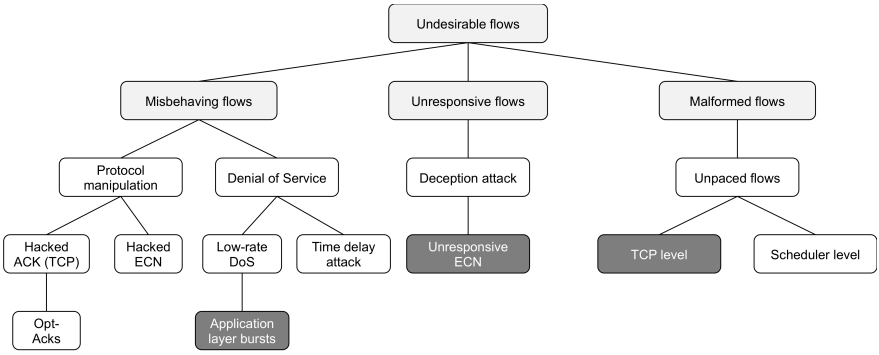


Fig. 1: Overall classification of the set of undesirable flows which may affect the proper delivery of a service in a networking architecture. Light grey boxes stand for the high-level categories of flows we consider in this paper, while dark grey ones for the scenarios we consider and further develop in the following.

3.1 Misbehaving Flows

Misbehaving flows are flows which do not respect the expected behavior of a considered service. They can occur due to both legitimate or malicious reasons. On the one hand, legitimate misbehaving flows are mainly due to interoperability issues of endpoint implementations which do not strictly conform to a protocol specification and consequently induce unexpected behaviors which are independent from the end-user intention. On the other hand, an attacker whose motivations may vary can deliberately modify the implementation of a protocol to take advantage of other participants. For instance, it can be a selfish objective which aims to benefit from more network resources (i.e. bandwidth, delay) than other end users, or a deliberate will to attack a service availability by degrading the quality of other legitimate flows without any actual consumption of this service.

In what follows, we only consider intended misbehavior, in other words, attack flows[3]. Misbehaving flows can be classified in two categories: protocol manipulation and DoS. A protocol manipulation is the ability of some of the participants to subvert the protocol without the knowledge of the others [19]. Most of these attacks are TCP-centered. We can first mention acknowledgement manipulation attacks, that we also call *hacked ACK* which manipulates a TCP endpoint to make the victim saturate the network (more specifically an edge router shared by the targeted victim). The optimistic acknowledgment (*opt-ack*) attack [19–21] is a well-studied example which consists in misleading a sender to send more packets. The receiver sends acknowledgements before it actually receives packets, leading the sender to behave as if the network was in good enough condition to send even more packets. A more discrete variant of this attack is the *lazy opt-ack* attack, which basically works similarly but the receiver conceals any packet loss by acknowledging all packets when only one may be actually received. Besides, congestion can be created in intermediate nodes by several manners. When it comes to manipulating ECN, we call it *hacked ECN*. One can conceal the congestion notification by not informing the remote entity of the communication [19, 22, 23]. An attacker can also generate false congestion notifications in order to steal more bandwidth. Some research for mitigating protocol manipulation has been presented recently in [21] in which the authors design an Extended Finite State Machine (EFSM) in the data plane using P4 to monitor and detect protocol misuses related to optimistic ACK and ECN abuse. The authors especially focused on detecting flows that are not protocol-compliant.

The other category of misbehaving flows concerns some forms of DoS. Despite the numerous studies and attempts to solve them, DoS attacks are ever a threat for novel networking architectures since their implementation can take different forms, adapted to the target intrinsic features. For instance, in a 5G context, Chen *et al.* [26] have identified a novel pattern of low volumetric DoS attack against both URLLC and eMBB (enhanced Mobile BroadBand) by

[3]However, legitimate misbehaving flows should not be ignored and should be addressed by the networking community in further research.

exploiting the vulnerability in the coexistence mechanism by a group of synchronized yet compromised URLLC UEs (User Equipments). More generally, Low-rate DoS attacks (LDoS) whose general model is described in [24] consists in sending periodic bursts of packets that are synchronized with the victim's Retransmission Timeout (RTO) in order to overflow the router's queue and eventually increase latency. LDoS attacks are more difficult to detect in comparison with volumetric DoS or DDoS attacks and can be sustained as long as the periodic generation of burst is appropriately synchronized, but they are consequently difficult to implement. Standing for another form of DoS attack, *time delay* attacks [25] aims at making cyber-physical systems unstable by delaying some signals to be received by some critical control nodes. This attack stands for an important threat against time-constrained systems due to the physical damages it can cause while being extremely stealthy and easy to implement [26]. As such, several works currently aim at bringing control systems with embedded protection systems such as [27] which proposes to integrate time delay attacks in perturbation estimation based control, which is an indirect estimation and compensation method to mitigate the effect of unknown system dynamics. Regarding L4S, the IETF has identified misbehaving flows as an issue [2, 10] for this architecture and it proposes a solution to some protocol manipulation, and more specifically for the case of ECN concealing whose main principle is as follows: the sender might set the IP-ECN flag itself instead of the router when a flow seems suspect. This way, a malicious receiver would have no idea whether the flag comes from the network or its remote peer.

3.2 Unresponsive Flows

An unresponsive flow is a flow that does not respond properly to congestion signaling (either ECN marks, dropped packets or additional delay). It can originate from any protocol that does not implement a congestion control such as legitimate UDP traffic or VoIP traffic. In the L4S context, both Classic and LL queues are subject to this kind of undesirable flows, which can lead to overloaded queues or to congestion signal saturation, thus, degrading performances. As mentioned by the IETF in [10], L4S can natively handle some unresponsive traffic, less responsive and/or temporarily unresponsive to congestion as long as its proportion is reasonable. However, it becomes an issue when it leads the DQC AQM into a queue-building behavior. To implement such a behavior, a malicious user can leverage ECN, with a scenario different from that exposed in Section 3.1, where he/she acts in a protocol-compliant way but, when congestion occurs, he/she sends the correct signalling to inform that he/she has reduced its rate while not doing so. This attack generates an unresponsive ECN-capable traffic and subsequently, we term it *Unresponsive ECN*. To the best of our knowledge, such a case of a misbehaving but protocol-compliant flow has not been covered yet and might be more subtle to differentiate from a legitimate flow. As such, as defined by Aggarwal *et al.* in [28], *Unresponsive ECN* can be considered as a form of *deception* attack

since it stands for *actions to promote the beliefs of things that are not true.* Deception attacks are currently extensively studied in cyber-physical systems [29–31] with a different scenario where spoofed sensors send wrong information to control systems in order to defeat the regulation performance or stability. In the case of a L4S, by sending ECN signals that do reflect the actual congestion rate, malicious endpoints aim at defeating the traffic regulation forming the core of the AQM control loop. To mitigate the effect of unresponsive flows in L4S, the IETF [2, 10] proposes some basic countermeasures which, however, induce to sacrifice some performances (L4S delay, L4S throughput or introducing L4S drop).

3.3 Malformed Flows

Malformed flows is the last category of threats for L4S we identified, which can degrade the performance of a networking system. It can for instance be the case of network devices sending burst of packets, which is mostly the case in cellular networks relying on transmission opportunities. As such, similarly to misbehaving flows, the main issue with this category of traffic is that the flow can be malicious, in case of attacks, or entirely legitimate due to characteristics as illustrated above with cellular network bursts. A solution to differentiate the intention of the user should be defined, but it is out of the scope of this paper.

The impact of malformed flows has already been considered in several networking architectures, such as in [32] which focuses on the impact of such flows[4] in a DetNet architecture. In this paper, the considered threat addresses the case where some packets arrive in a Time Aware Scheduler out of their reserved time slot. In that situation, the authors propose some mechanisms to prevent the scheduler to enqueue them, eventually leading to congestion for subsequent packets arriving during their reserved time-slot.

From the L4S perspective, a bursty behavior may occur when the network stack of regular operating systems within endpoints waits for the sending buffer to be filled before sending data over the network. This results in on/off patterns hurting the L4S performance. Such a pattern of DualPI^2 overloading has been studied in [33]. This study shows that under overloading traffic, the response time of the PI controller depends on the buffer size of the router. A large buffer can reduce on/off emitting patterns generated by a sender while a small buffer can better restrict the queuing delay at the price of tail dropped packets. However these experiments were made with DCTCP and UDP overloading traffic without the Prague congestion control nor with a malicious user. Finally, in [4], the performance of L4S architecture has been proven to be sensitive to the flow burstiness induced by the Linux kernel (e.g. segmentation offloading or pacing setting). We confirm this observation in Section 5.3 and show that Prague pacing mitigates this effect. The authors tested TCP Prague on DualPI^2 and showed that it performs a better sharing

[4] In [32], the authors refer to this type of flow as *misbehaving* ones, which is not compliant with the taxonomy proposed in Figure 1. However, the authors neither relate their work to any flow classification nor motivate this naming as compared to other types of undesirable flows.

behavior than DCTCP. However, the L4S performance has not been studied in a context where a malicious user is willing to exploit the effect of sender burstiness sensitivity to increase its impact on other legitimate flows. Besides, traffic shaping and traffic policing (or queue protection) are also considered as a solution to deal with malformed flows [2, 10]. However, classical techniques for traffic shaping are not straightly applicable in a LL context, as they may lead back to the bufferbloat problem. TCP pacing is a solution that can be required for endpoints to respect before sending traffic on the network. When combined with fair-queuing within the endpoints network schedulers, it can drastically reduce traffic (micro-)burstiness.

Problem Statement: As seen in the previous sections, attacks targeting L4S can occur and some threats have already been identified by the IETF. However, to the best of our knowledge, there is neither any comprehensive study of their impact on LL traffic nor any implementation of dedicated detection and mitigation components such as those cited above, thus making the case of undesirable flows handling in L4S an open issue to date. As such, in this paper, we do not present novel attack scenarios but we propose to fill this gap by performing an exhaustive evaluation of the attacks' impact on the L4S architecture. Properly characterizing those impacts is indeed critical for L4S in order to efficiently define detection mechanisms and counter-measures to mitigate them and eventually facilitate its adoption and deployment.

4 Experimental setup

To understand and evaluate comprehensively how the L4S architecture behaves when some of the identified undesirable flows happen and what the possible impacts on a legitimate LL flow are in a realistic deployment, we consider a real end-to-end scenario that simulates Internet-like network conditions, where both a legitimate and malicious users share a common access network implementing a L4S router to reach some services located in a different network and made accessible through the Internet. This deployment scenario differs from those previously implemented in [4, 33] since the purpose of the present study is not to benchmark the L4S performance but rather to assess to what extent some particular malicious flows may threaten the good delivery of legitimate ones in a realistic deployment.

4.1 Testbed

As depicted in Figure 2, we implemented a testbed composed of virtual machines for clients and servers running under OpenStack Juno cloud platform (v2.19.0 - 2014, using linux kernel 3.13), and a baremetal server acting as a router. The baremetal router is equipped with an Intel(R) Xeon(R) CPU E5-2430 v2 @ 2.50Ghz, and two Intel Corporation Ethernet 10G 2P X520 adapters. The baremetal router runs DualPI^2 - the reference implementation

of the Dual Queue Coupled AQM proposed by the IETF - as an egress queuing discipline on the interface of the client side, referred to as *eno1*. Other interfaces are using *pfifo_fast*. All nodes, including the baremetal router, are synchronized with the *chrony* v2.1.1 NTP client implementation.

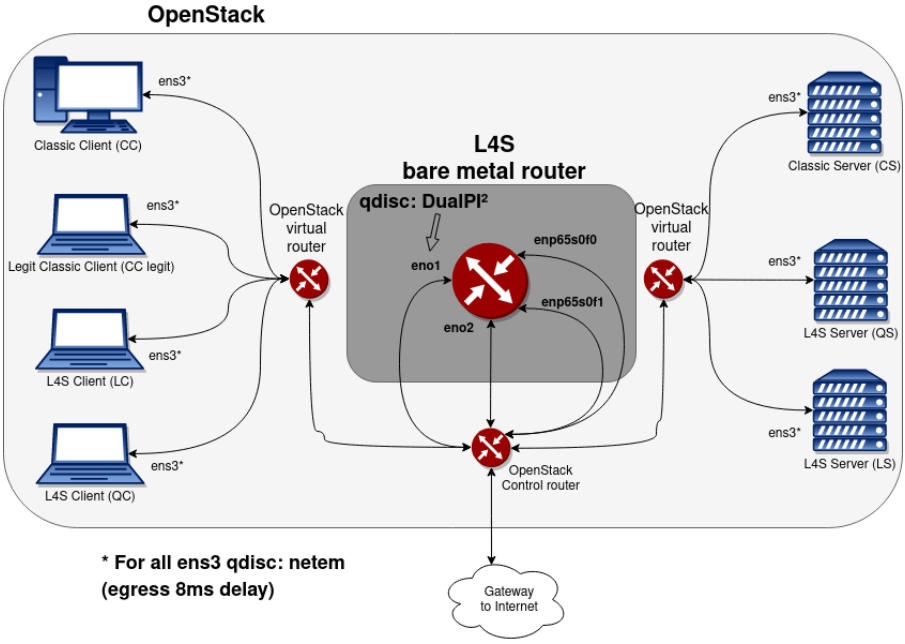


Fig. 2: Testbed leveraging OpenStack as a cloud computing platform interacting with a baremetal router that implements DualPI² on its client-side interface (*eno1*)

Senders and receivers can be Classic (C), low-latency over TCP (L), low-latency over QUIC (Q), and they can be Client (C) or Server (S) as shown in Figure 2. L4S sender and L4S receiver nodes are using the Linux kernel 5.10 from the L4STeam repository[5] and are configured to use ECN with the convenient codepoint to be classified in the low latency queue. Receivers are connected to the same router interface (*eno1* in the figure), Classic senders and low latency senders are connected to separate interfaces. We can adjust *maxrate* and enable or disable layer-3 pacing with the system program traffic control (*tc*). A one-way delay of 8 ms is introduced with *netem* as an egress queuing discipline on the interface of servers and clients, which makes a total of 16ms of base delay as a RTT, in order to simulate real Internet conditions.

On the egress direction of *eno1* (receivers side), the router implements DualPI² which is tuned with a configurable maximum rate (1 - 999 Mbps), all

[5] https://github.com/L4STeam/linux

other parameters are left to default, i.e. 10000 packets limit, the coupling factor k is set to 2, *drop_on_overload* is the strategy to adopt when high congestion occurs, the target queue delay is 15 ms for the Classic queue, aggregated packets are split with the *split_gso* option and the step threshold, in other words, the sojourn time threshold from which DualPI^2 will always mark exceeding packets within the low latency queue, is set to 1 ms. Finally, in order to determine the throughput limit of our testbed and ensure that it would not be the source of a bottleneck for our subsequent experiments, we ran *iperf3* between our L4S senders and receivers without limiting the router rate, and reached a throughput limit around 850 Mbps.

4.2 Traffic Generation, Data Acquisition and Processing

The legitimate flow, generated with *iperf3*, is controlled by TCP Prague while the malicious node uses QUIC, a protocol based on UDP, which is enhanced in the user space with congestion control algorithms and other features that are present in TCP. This lets a malicious user the ability to modify the sending behavior easily without requiring root privilege to tune or recompile the Linux kernel networking stack and as such it stands for a realistic scenario to consider. We especially chose to use *picoquic*[6], a minimalist implementation of QUIC compliant with the Draft-17 [34] version of QUIC, which has been forked by the L4S team[7] to support the Prague congestion control.

The monitored metrics are collected from the endpoints with two different methods according to the source of traffic (TCP via *iperf3* and QUIC via *picoquic*). For TCP-generated traffic, metrics are collected by calling socket statistics (*ss*). This provides the tcp_info data structure of the kernel, which contains the reported RTT and its mean deviation, the congestion window (*cwnd*), the maximum potential sending rate based on the formula: $\frac{cwnd \times MSS}{RTT}$ and last bytes acknowledged.

For the router, the monitored metrics are collected using *tc*, which provides metrics from DualPI^2. These are the Classic and low latency queue occupancy and queuing delay, the marking probability from the PI controller (base probability), the number of ECN marks and the number of dropped packets.

4.2.1 Instrumentation Impact

We evaluated the impact of our instrumentation on the traffic we aim to characterize. We first assessed the impact of the sampling rate of the router on the flow occupancy by letting a standalone *iperf3* flow pass through the router. We tested various sampling rates from 1 sample each 4 RTT to 4 samples per RTT and for two bounding rate limitations of 10 and 100 Mbps in the router, standing for those we consider in subsequent experiments. The results, depicted in Table 1, show that the flow occupancy is almost constant whatever the throughput and measurement frequency we consider. We have actually

[6] https://github.com/quicwg/base-drafts/wiki/Implementations#picoquic
[7] https://github.com/L4STeam/picoquic

encountered some impacts of the router instrumentation on flow occupancy but on a range that is out of the scope of that we selected in this paper (i.e. 500 Mbps for the router throughput with more than 20 measurements per RTT leading to around 60% of flow occupancy, which means 40% of unused bandwidth). As such, we collect four samples per RTT in our different experiments for both the router and the *iperf3* traffic, making the sampling rate not impacting the flow performance and thus preventing any bias in the presented results.

Router throughput	1 measurement per 4 RTT	1 measurement per RTT	4 measurements per RTT
10 Mbps	94,60%	94,70%	95,60%
100 Mbps	91,35%	91,26%	93,49%

Table 1: Achieved flow occupancy of a legitimate TCP *iperf3* traffic for different router throughputs and sampling rates. Flow occupancy evaluates to what extent our instrumentation introduces a performance impact.

Similarly, we also evaluated the *picoquic* logging type on the flow occupancy to see if our instrumentation degrades the traffic performance. We ran a *picoquic* flow with a 10 and 100 Mbps rate limitation on the router in different logging situations: logging disabled, native *picoquic* logging enabled, and finally, using our instrumentation. We observed no specific performance issues whatever the logging type.

4.2.2 Undesirable Traffic Injector Performance and Limits

As the version of *picoquic* used to implement the Prague congestion control algorithm follows the draft 17 of QUIC [34], it focuses on the correct implementation on the protocol features rather than its performance in terms of capability to support a high throughput. This leads us to evaluate traffic generation variability to determine the range of throughputs that does not introduce substantial bias in the traffic generation due to implementation limits while being compatible with the network conditions we aim at reaching to highlight the impact of undesirable flows on a legitimate traffic. To that aim, we evaluated the *picoquic* performance against that of *iperf3* for throughputs ranging from 1 to 100 Mbps as a maximum rate limited by the router and for a base RTT of 1 ms. We ran a set of 10 tests for each situation in which we downloaded a randomly generated file equivalent to half of the router rate (e.g. a 5 MB file for 10 Mbps and a 25 MB file for 50 Mbps). To be able to transmit files above 1 Mo, we tuned two pre-processor variables[8], thus creating a larger buffer to be able to randomly generate larger files. Figure 3 depicts the median, first and last quartile of the flow occupancy, standing for the throughput reachable by *picoquic* against the router throughput limitation. It shows that the *picoquic* flow occupancy decreases starting from a router throughput

[8]Respectively `H3ZERO_RESPONSE_MAX` and `PICOQUIC_FIRST_RESPONSE_MAX`

of 20 Mbps and eventually collapse up to a flow occupancy that cannot exceed around 40% for 100 Mbps. We can also notice that some outliers are visible, due to the first connection, which always benefits from a better throughput as compared to all subsequent ones. Eventually, this experiment exhibits a relevant variability of results related to some unpredictable behavior of *picoquic* for these throughputs. Despite these phenomena, *picoquic* is, however, the only QUIC implementation of the Prague CCA currently allowing to compete as an attacker, without any root access on a machine, with a legitimate flow in the LL queue while being able to disturb some mechanisms (e.g. pacing, ECN responsiveness) at application-layer level. As such, we selected it with an enlightened choice of the main throughput we choose for the subsequent results which are set to a 10 Mbps value, acting as the right trade-off between throughput and reliability.

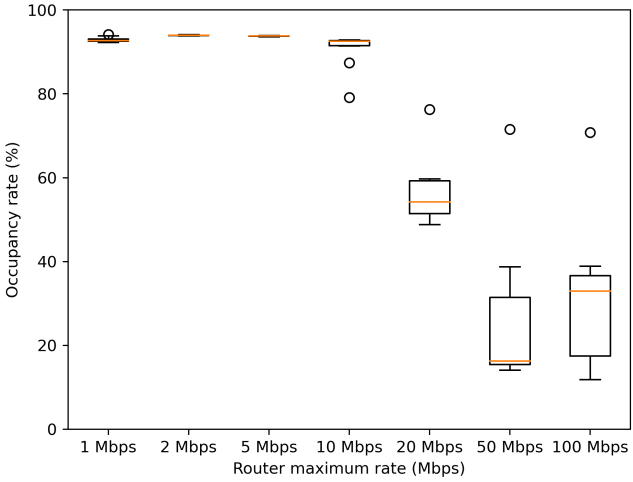


Fig. 3: Flow occupancy of *picoquic* for different router throughput limitations

4.2.3 Data Processing Methodology

Each experiment lasts 60 seconds and for each, we select the time period out of 60 seconds that best illustrates the phenomenon we want to observe, and removes all the artifacts, especially (1) those at the beginning of experiments, such as the unpredictable behavior of a first *picoquic* connection as explained in Section 4.2.2 and (2) periods of time when the *iperf3* traffic is solely present. To that aim, we apply the following methodology. During the 60 seconds of each experiment, we launch *picoquic* traffic twice and select only the data corresponding to the second shot. The duration of the *picoquic* traffic varies depending on the router rate limitation and the size of the file to download. Consequently, the duration of the phenomena presented in the rest of the

paper, including Section 5 and Section 6, uses the same time window selection in order to clean the data from noise.

4.3 Time-series analysis of the Reference Traffic

In order to highlight and quantify the impact of undesirable flows on legitimate ones, we set up a control sample acting as a reference to compare with our different scenarios. To that aim, we consider that the measurements depicted in Figure 4 act as the baseline of our experiments. It shows regular network conditions when our testbed carries one TCP low latency flow and one unaltered QUIC low latency flow. The reader can refer to it so as to identify the traffic degradation due to the different undesirable flows we consider subsequently.

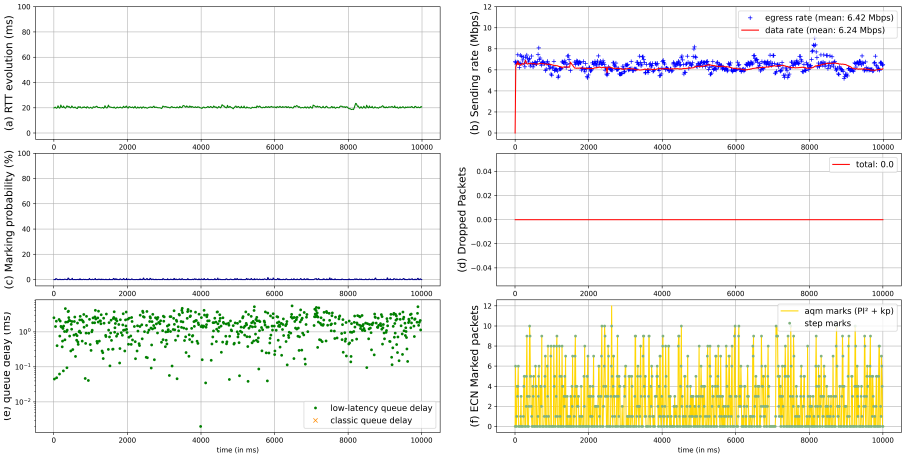


Fig. 4: Time series of the standard behavior of the L4S architecture with one TCP low latency flow and one unaltered QUIC low latency flow (router rate is set to 10Mbps). (a) TCP RTT evolution (ms). (b) TCP Sending rate (Mbps). (c) Marking probability (%). (d) Dropped packets. (e) Queue delay (ms). (f) ECN Marked packets. Horizontal axis is the time in ms.

In this reference situation, the router sends a maximum of 10 Mbps, shared between those two flows. The RTT of the TCP flow is represented in Figure 4a on which one can observe that it is stable and close to the base RTT set to 16 ms in all our experiences. Figure 4b, representing the sending rate of this TCP flow, let us deduce that the QUIC flow is taking a bit less that half of the available bandwidth. The maximum potential sending rate is depicted in blue "+" and the red line indicates the actual data rate, based on received acked bytes. Figure 4c shows the base probability of the Classic queue which constantly remains very low, thus indicating a low level of congestion over the entire experiment duration. This is further confirmed by Figure 4d which shows that no packet drop is necessary to guarantee the flows respective requirements. Looking in a deeper way, Figure 4e shows queue delay for both flows

ranging approximately between 0.03 ms and 6 ms, with a mean centered on 1 ms, reflecting the expected traffic regulation enforced by marked packets, represented in Figure 4f. For LL packets, the decision to mark packets is based on the maximum between the probability of LL queue, which eventually corresponds to the number of step marks also shown in Figure 4f in green dots, and the base probability of the Classic queue, shown in Figure 4c, multiplied by the coupling factor k.

5 Time-series analysis of Undesirable Flows

This section explores to what extent a legitimate LL flow is affected by an unexpected behavior in the presence of undesirable flows. We consider one scenario per type of undesirable flow identified in section 3 and, in order to characterize each of them, we analyse their time series with a set of metrics identical to those considered to characterize the normal behavior of L4S as detailed in Section 4.3.

5.1 Unresponsive ECN

As an unresponsive flow, we first propose to implement an ECN abuse attack targeting the increase of a legitimate flow latency. The protocol manipulation we implemented consists in being unresponsive to congestion notification while respecting ECN signaling. This was achieved by removing the congestion window reduction when updating the coefficient of reduction in Prague as well as preventing the New Reno recovery mode to reduce the slow start threshold. As such, the expected consequence is the saturation of the LL queue, which in return will increase the marking probability (leading eventually to some packet drops) and thus generating an extra delay, sufficiently high to make the LL application unusable. A side effect of this attack relies in bandwidth stealing: such a behavior will indeed take advantage of the reduction of other participants when a congestion event occurs, resulting in the robbery of all the available bandwidth for legitimate users.

Figure 5 shows the time series for the different metrics we consider in the context of a malicious user generating such an unresponsive ECN-capable flow. We can see that it brings a higher queue delay which in return increases the average RTT. The RTT increases progressively until reaching 80 ms, which is four times higher than the baseline plotted in Figure 4. Unresponsive ECN puts DualPI^2 to saturation and triggers the *drop_on_overload* reaction. The number of ECN marks is growing up to a stable phase, which corresponds to the moment when there is no additional bandwidth to steal. In this stable phase, the number of ECN marks is twice bigger than step marks (i.e. amount of ECN marks due to the threshold exceeding). The number of step marks is around 10 per consecutive measure points, compared to Figure 4.f which was 0 most of the time, or between 1 and 4 else. We can also notice that the marking probability is around 50% after the growing phase, yet it is rather stable. The sending rate is also stable yet very low. This is due to the fact that the

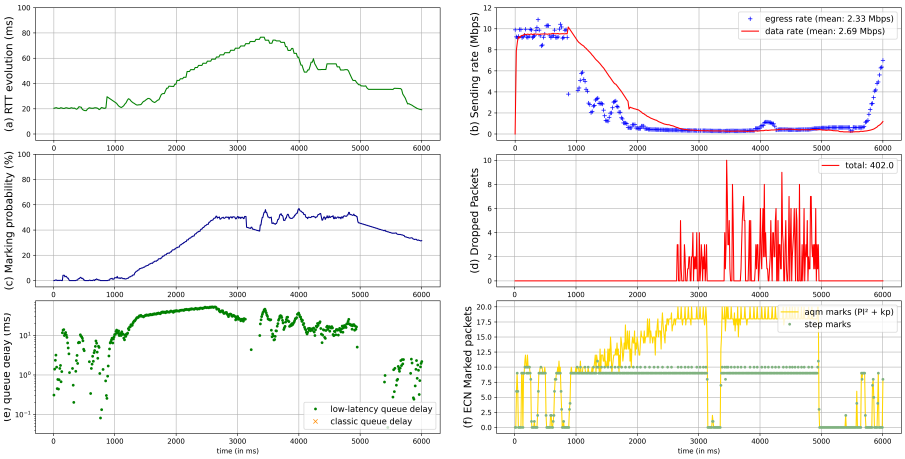


Fig. 5: Time series of impacts of Unresponsive ECN on a legitimate low latency flow (router rate is set to 10Mbps). The vertical axes of the subfigures are the same measured metrics used for subfigures of the standard behavior (the first two show metrics from the legitimate endpoint, the others show metrics from the router) and the horizontal axis is the time in ms.

legitimate flow responds correctly to congestion notification while the attacker steals all the available bandwidth pretending to reduce its sending rate.

5.2 Application Layer Bursts

In this section, we address two questions regarding the L4S vulnerabilities against undesirable traffic. The first one deals with the impact of a misbehaving flow on the low latency legitimate traffic. As such, we generate traffic bursts of short client requesting 80 ko files over QUIC. The second one deals with the capability of a Classic sender to disturb a legitimate flow in the low latency queue and thus exploit the coupling of the queues as an attack vector. To that aim, we configure the ECN flags of the attack traffic to be classified in the Classic queue. The bursty behavior is expected to trigger a high queue variation which is more likely to increase the marking probability due to the PI proportional gain β (see Equation 1) in comparison with a constant saturation.

Figure 6 shows the results we collected with such a malicious user generating a bursty flow within the Classic queue. One can notice that in Figure 6e, the green points represent the LL flow queue delay while the orange "x" represent the attacker delay within the Classic queue. The marking probability, 6c, is erratic but around 20% in average with an oscillation of 20% amplitude. ECN marks follow the shape of the marking probability, which is something expected, and occur ten times less often than in case of Unresponsive ECN. Step marks are negligible, even compared to Figure 4. This is due to the presence of a single flow in the low latency queue. In other words, the potential

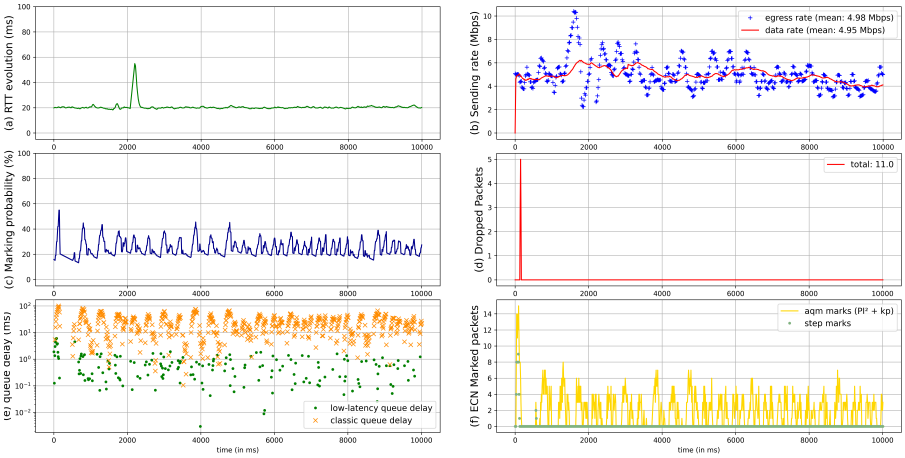


Fig. 6: Time series of impacts of application layer bursts within the Classic queue on a legitimate low latency flow (router rate is set to 10Mbps). The vertical axes of the subfigures are the same measured metrics used for subfigures of the standard behavior (the two first show metrics from the legitimate endpoint, the others show metrics from the router) and the horizontal axis is the time in ms.

queue building behavior is only due to this particular flow, as there is no concurrency to fill the buffer. The variation in the rate of the legitimate flow is due to the coupling mechanism in DualPI^2 which is designed not to hurt Classic flows since it is rather preferred to sacrifice L4S delay on saturation to ensure a fair flow coexistence. More precisely, when the LL queue delay is under 1 ms (which is the case most of the time here when we look at the number of step marks), the marking probability of the LL queue is governed by the base probability multiplied by the coupling factor k. This is why, in the end, the sending rate of the legitimate low latency flow is affected. To conclude, RTT and low latency queue delay are only affected when the first burst occurs, showing that the DualPI^2 AQM seems to protect the low latency traffic correctly when the burst takes place in the Classic queue. Consequently, in order to determine if a bursty flow in the LL queue would bring more impact to the latency of the LL legitimate flow, we have conducted other experiments under this latter configuration, which showed that the LL legitimate flow becomes reasonably affected when the burst goes within the LL queue with a RTT oscillating from 20 ms to 30 ms.

5.3 Transport Layer Unpaced Traffic

A solution to avoid bursty emitting patterns to be sent from an endpoint is to pace packet emission over a RTT instead of sending a bulk. This operation can be accomplished by the TCP transport layer of the Linux kernel and also as a queuing discipline with Fair Queuing (FQ). Previous work [4, 33]

mentioned that when FQ pacing (i.e. kernel-level pacing) was disabled, the pacing within the Prague congestion control algorithm was good enough to alleviate burstiness of the traffic sending patterns.

In this experiment, we used $pfifo\_fast$ as a queuing discipline at the sender endpoint, which lets us control the pacing on the egress traffic only from the transport level. When pacing is disabled, it generates undesirable flows. This kind of undesirable flow is meant to generate micro-saturation with sub-RTT bursts, also termed microbursts. Considering such a vulnerability of L4S as identified in the literature at the kernel level which may not be easily tunable, we aim at evaluating to what extent disabling Prague pacing within the user space, through the QUIC stack, exhibits an impact and can be of a potential interest for an attacker to generate easily microbursts hence defeating the LL feature of the legitimate flow.

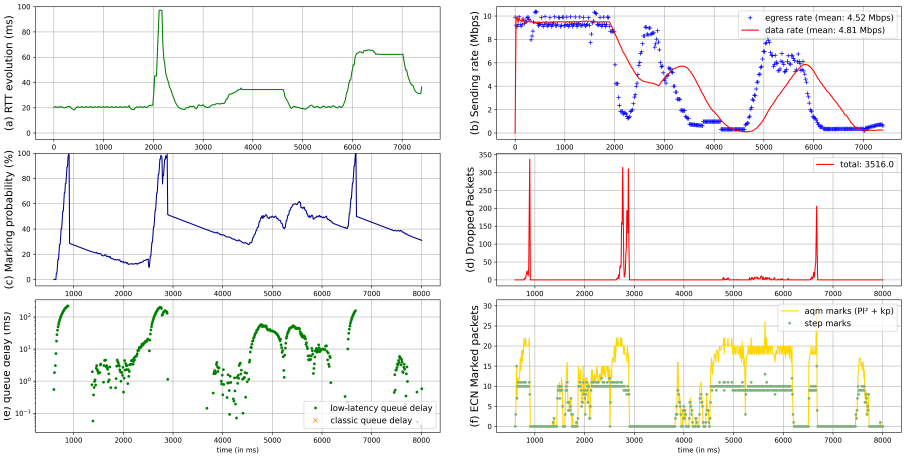


Fig. 7: Time series of impacts of a malformed (unpaced) flow on a legitimate low latency flow (router rate is set to 10Mbps). The vertical axes of the subfigures are the same measured metrics used for subfigures of the standard behavior (the first two show metrics from the legitimate endpoint, the others show metrics from the router) and the horizontal axis is the time in ms.

Figure 7 shows the results we collected in the context where the malicious user is generating such microbursts. As we can observe, time granularity of the measurements does not allow to see microbursts directly. Instead, we see the consequences of an indistinguishable aggregation of packets. The time scale is too high to properly observe the source phenomenon. Moreover, this phenomenon is propagated in forwarding equipments, such as OpenStack switches and routers, and may induce additional phenomena that we do not control. Yet, we still can study microbursts through their consequences. Figure 7c show that the marking probability is less erratic compared to the previous experiment, but still not stable and presents large spikes almost reaching 100% of

probability. This metric, in stable phases, is in average around 40%, which is similar to the Unresponsive ECN case. This leads to a large number of ECN marks, corresponding to the saturation phase in the Unresponsive ECN experiment, with a similar number of dropped packets. RTT delay shows a spike almost reaching 100 ms at 2 seconds. More specifically, we can observe that it alternatively goes back to normal and then switches to different plateau (35 ms at 4 seconds and 62 ms at 6 seconds). The sending rate has the widest dispersion among others measured undesirable flows of our study. We can see that it is mostly due to LL queue delay which also has the widest dispersion, reaching about 200 ms as a peak value and exhibiting a strong instability. The sending rate is higher than for the Unresponsive ECN scenario because the legitimate flow is able to increase its congestion window between microbursts.

As anticipated previously, the marking probability is very sensitive to high variation due to the weighting factor β of the PI controller. Thus, when a burst occurs, DualPI^2 reacts very quickly to punctual events, resulting in a sending rate reduction from the legitimate user which, in the end, induces heavy fluctuations and wide dispersion of sending rate possible values.

6 Impact Characterization of Undesirable Flows

To further explore the previous results and reveal some insights that can help the design of a detection system, we have (1) extended the set of previously exposed scenarios to other network conditions and (2) applied a Principal Component Analysis to some selected experiments in order to reveal the most meaningful metrics in each situation.

6.1 Global statistical analysis

In order to characterize exhaustively the impact of undesirable flows according to different factors, we extended the snapshots of measurements described in the previous section to a comprehensive campaign encompassing 30 different situations. We have particularly studied to what extent undesirable flows impact a legitimate low-latency one according to the router rate limitation (with values within the range from 10 to 100 Mbps) and for three ECN configurations of the *picoquic* flow: no ECN at all (thus making the congestion phenomenon being notified to endpoints through packet drops), ECN with the ECT(0) codepoint (corresponding to Classic traffic), ECN with the ECT(1) codepoint (corresponding to low latency traffic).

We reported in Table 2 the essential statistics which allow to understand to what extent an undesirable flow impacts the legitimate one. These are the mean and variance values of the RTT, the low latency queuing delay (LQ delay) and the legitimate TCP flow rate. The construction of the labels used to differentiate each row is summarized in the caption and can be decomposed as follows: the first part identifies which kind of experiment is made, the second one identifies to which queue the *picoquic* traffic is inserted and the last part is the rate limitation (in Mbps) at the router egress rate.

Scenario	RTT mean	RTT var	LQ delay mean	LQ delay var	TCP rate mean	TCP rate var
#1: L_NE_10M	19.594	0.247	0.317	0.469	7.456	0.659
#2: L_NE_20M	18.472	0.416	0.43	0.445	17.094	0.808
#3: L_NE_100M	17.207	0.541	0.095	0.226	77.014	5.138
#4: L_C_10M	19.649	0.343	0.374	0.527	7.563	0.737
#5: L_C_20M	18.482	1.795	0.351	0.407	16.284	1.461
#6: L_C_100M	17.285	0.338	0.31	0.339	83.721	3.99
#7: L_LL_10M	20.428	0.575	1.033	1.152	6.425	0.516
#8: L_LL_20M	18.463	1.158	0.458	0.806	16.737	1.541
#9: L_LL_100M	17.539	0.337	0.451	0.325	90.781	1.953
#10: B_NE_10M	20.113	0.557	0.108	0.356	5.087	1.28
#11: B_NE_20M	18.477	0.246	0.073	0.198	10.503	0.796
#12: B_NE_100M	17.133	0.16	0.011	0.061	53.49	4.748
#13: B_C_10M	20.426	2.598	0.151	0.535	4.981	1.244
#14: B_C_20M	18.452	0.331	0.085	0.217	11.09	2.146
#15: B_C_100M	17.077	0.134	0.014	0.081	55.266	5.328
#16: B_LL_10M	21.049	2.649	2.039	3.628	7.298	1.167
#17: B_LL_20M	18.322	1.041	0.49	1.189	15.937	1.246
#18: B_LL_100M	17.435	0.348	0.368	0.332	90.354	2.358
#19: M_NE_10M	20.386	0.553	0.135	0.444	9.502	0.448
#20: M_NE_20M	19.137	2.218	0.054	0.201	9.023	6.203
#21: M_NE_100M	19.043	1.884	0.002	0.012	9.6	7.346
#22: M_C_10M	21.02	2.055	0.175	0.535	4.85	3.366
#23: M_C_20M	18.578	1.687	0.092	0.207	9.71	4.887
#24: M_C_100M	17.74	0.966	0.105	0.252	46.337	33.182
#25: M_LL_10M	32.083	16.563	19.131	41.087	4.522	3.734
#26: M_LL_20M	28.014	25.018	15.187	40.292	9.699	5.183
#27: M_LL_100M	26.681	29.087	11.477	31.473	48.881	28.487
#28: U_LL_10M	43.592	18.419	18.528	16.749	2.326	3.247
#29: U_LL_20M	21.086	3.502	2.253	4.863	8.069	5.502
#30: U_LL_100M	33.644	24.649	9.068	23.487	7.342	6.196

Table 2: Mean values and standard deviations of main metrics (RTT, Low-latency Queueing (LQ) delay and TCP rate) of the full data set, with the following legend of scenario names: L: Legitimate trafic (i.e. unaltered *picoquic* flow); B: Bursty flow; M: Malformed (unpaced) flow; U: Unresponsive ECN; NE: No ECN support; C: Classical queue; LL: Low latency queue.

Rows #1 to #9 exhibit the case of two flows, one over TCP with *iperf3* and the other with an unaltered version of *picoquic*, thus standing for legitimate flows only. Row #7 is, for instance, the synthesis of the scenario we depicted in Figure 4 and we detailed in Section 4.3. Overall, in this legitimate scenario, the RTT is very stable in general with a mean value between 19 and 20 ms over the different router rates. The situation in which the two flows share the LL queue leads to a small increase in the queuing delay, which is an expected phenomenon, with a lowest value of 0.3 ms (row #6) in case of separated queues, up to 1 ms (row #7) when all the traffic passes through the LL queue. The overall performance is slightly improved when the router rate increases

(all rows except #30), which is expected as the congestion point is further postponed as the rate limitation is released.

However, despite their common implementation of the Prague congestion control algorithm (CCA), our two sources of legitimate traffic do not operate in a fair way. Indeed, the bandwidth is not equally shared, as we can see when we compare the mean TCP rate with the rate limitation indicated in the label of the scenario. For 100 Mbps, for instance, the *iperf3* flow captures from 75% to 90% of the available bandwidth as the router rate increases(rows #3, #6, #9). This is explained by the implementation limitations of *picoquic* highlighted in Section 4.2.2 which show that, even in a standalone way, the QUIC flow cannot fully occupy the allocated resource by the router.

Application layer bursts results, synthesized from row #10 to #18 clearly state that impacts of such type of misbehaving flow cannot be measured with mean values, as spikes are not taken in consideration in the mean RTT of the legitimate flow. For a router rate of 10 Mbps, one can, however, notice an increase in the RTT variance growing up to 2.6 ms (row #16) and thus denoting a relative instability in the latency which may start to be an issue for services where jitter has to be bounded. This is further confirmed with the low latency queue (LQ) delay variation when the two flows pass through the LL queue (row #16) with a value of more than 3.6 ms, exceeding the mean of 2 ms. Besides, the main impact of the bursty flow is related to the TCP throughput (rows #10-#15) which collapses from more than 75% of flow occupancy for the legitimate case to 50% with such a type of undesirable flow going through the Classic queue, thus exhibiting here a potential negative impact of the coupling factor of the DCQ AQM. By contrast, when the bursty flow passes through the LL queue (rows #17-#18), the legitimate flow does not seem to be altered, with values in the same order of magnitude as those of the case of the legitimate scenario. This can be explained both by the capacity of scalable congestion control to quickly reach the operating point after an event of congestion and by the fact that application-layer bursts of a single flow may not be powerful enough to impact the legitimate flow in this configuration.

The case of malformed (i.e. unpaced) flows (rows #19 to #27) exhibits different yet worst impacts as compared to application layer bursts. When the malformed flow passes through the Classic queue, the latency of the LL flow is preserved with no significant impact on the LL mean and variances. However, considering the case of a malformed flow cohabiting with the legitimate one in the LL queue, one can see a relevant negative impact on both the legitimate flow latency mean and variance (rows #25-#27). This situation is not a discovery since it was especially identified in [4] but we confirm here that, with the deactivation of the transport level traffic pacing in the QUIC stack, one can easily and strongly perturb the expected behavior of legitimate coexisting LL flows. Such a behavior also impacts the throughput of the legitimate flow in an unexpected way due to the extremely erratic behavior of the *picoquic*

traffic. Here, whatever the considered queue for the malformed flow, the available throughput for the legitimate flow cannot exceed around 50% in the best cases, thus exhibiting a strong impact on the legitimate LL traffic.

Finally, the case of unresponsive flow (rows #28 to #30), which does not make sense if not implemented in a direct concurrency with the LL one, is clearly the one exhibiting the strongest impact among all our experiments. In particular, for a router rate of 10 Mbps (row #28), the *picoquic* flow manages to strongly degrade the LL latency behavior with a LQ delay of 18 ms on average and a roughly identical variance, while almost fully starving the available bandwidth of the legitimate LL flow. The impact on the latency, however, reduces when the router rate increases for the same reasons explained in the legitimate case, but the impact on the flow rate remains huge with more than 90% of the starved bandwidth.

Limitations: Throughout this large consideration of different flow coexistence situations, one can clearly highlight the impact of different undesirable behaviors implemented over different configurations of our testbed. The main limitation which needs to bring some caution in the exposed results relies in the limited performance of the *picoquic* implementation we considered in our experiments which seems to support hardly throughput increases as motivated in Section 4.2.2 in Figure 3 and reminded here. Since the main purposes of our study are on the understanding of the impact of non-standard traffic and assessing to what extent such flows can threaten the LL expected features, we have deliberately chosen to use *picoquic* without attempting to enhance its throughput support. Further work in that direction could be useful to refine the results of the present paper, especially in terms of bandwidth fairness and flow pacing when the router rate increases. However, given the results collected in the case of both legitimate flows, whatever the router rate, we show that the limitations of *picoquic* do not impact the legitimate flow operation since it follows a behavior compliant with L4S scalable congestion control algorithm.

6.2 Principal Component Analysis

In order to understand better our dataset and especially how undesirable flows impact the various metrics, we have applied a Principal Component Analysis (PCA) for each type of undesirable flow. This can be useful in order to design statistical methods in the perspective of detecting those threats. PCA is a statistical analysis technique used to reduce the number of dimensions of a dataset, hence easing its representation and manipulation, while maximizing the preserved amount of information. To that aim, we took the same set of experiments, as those presented in Section 5 except for the misbehaving flow, for which we analyse the impact within the low latency queue instead. For each experiment we used the router metrics described in 4.2. PCA belongs to the category of *unsupervised* methods because the label of data is not used. Instead, taking into account the covariance between the metrics, we represent them using principal components that keep as much information as possible

(F1, F2, ...) instead of the original space of metrics (LQ delay, marking probability, ...). Each principal component is defined as a linear combination of original metrics, representing their contribution. In addition, each principal component is associated with the percentage of information it provides on the whole dataset. Therefore, one would naturally rank the principal component in descending order to select the minimal number of components required to achieve a desired degree of amount of conserved information. In other words, the information contained in the dataset can be expressed with a certain percentage of loss, depending of the number of principal components considered (if we consider every principal component, there is no loss, but also no dimension reduction).

As such, in the following, each undesirable flow will be studied with two representations:

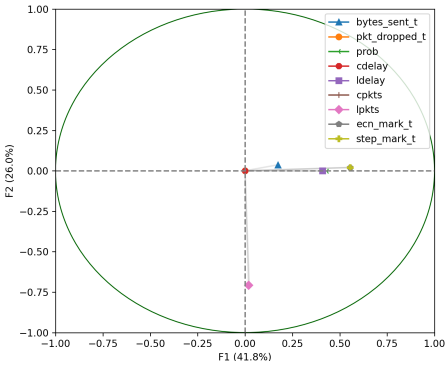
- A correlation circle which is a projection of each metric on the plane defined by the first two principal components. To understand better this representation, one should know that the length of each symbol represents how much each the associated metric matters when explaining main principal components. Also, the projection of the symbol onto x and y-axis represents how much the considered principal component is defined by the metric associated with this symbol.
- A projection of individual measurement points on the first two factorial planes, to highlight the evolution of distinguishable behaviors for better understanding them, especially with the help of correlation circles on what the principal components are made of.

In Appendix A, bar diagrams are provided to assess more finely the composition of principal components when metrics are too close in their correlation circle. The reader can refer to it especially where the correlation circle is described.

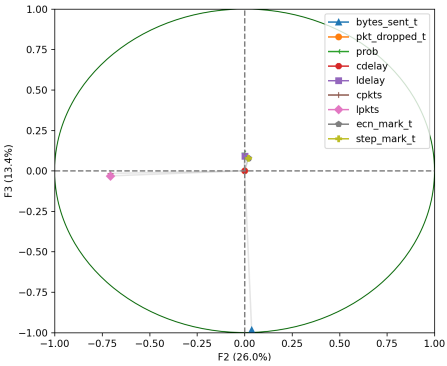
6.2.1 Legitimate case

In the scenario with only legitimate users, more than 80% of the information contained in the data are captured with the three main components, the first one accounting for 41.78%. We can understand the first component F1 as the normal behavior of a low latency flow passing through DualPI^2: the LL queue delay is increasing proportionally following probability and marking, which are operating by proactively preventing a congestion happening. When looking at Figure 8a one can observe that this first principal component is mostly made of the marking rate (step marks or simple ECN marks), the base probability and, to a lesser extent to the low latency queue delay.

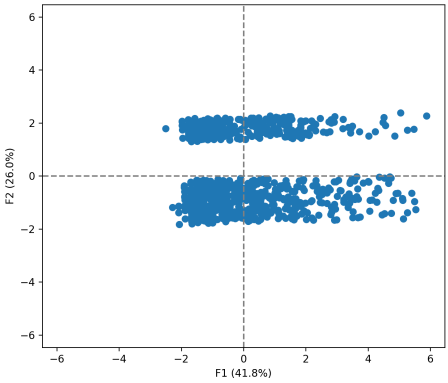
The second component F2 represents 26.0% of the information and represents only the queue occupancy, for both queues. F2 can be understood as a variable that measures queue emptiness or queue availability. The third component F3, visible in 8b, represents 13.4% of the information and also almost exclusively represents one metric, the number of bytes sent and slightly. F3 is



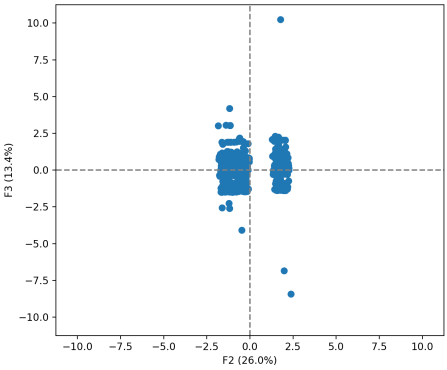
(a) Correlation circle for F1 and F2



(b) Correlation circle for F2 and F3



(c) Projection of measurement points in the first factorial plane (F1 and F2)

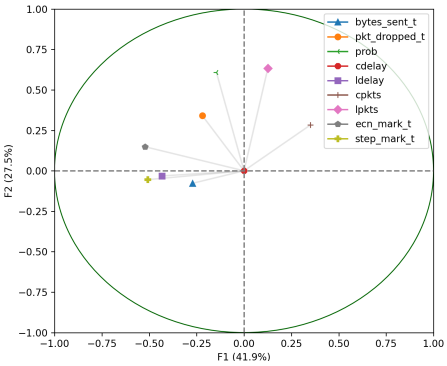


(d) Projection of measurement points on the plane directed by F2 and F3

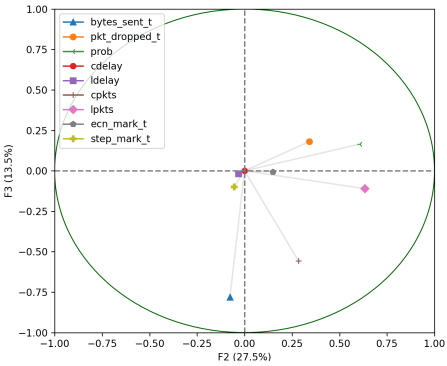
Fig. 8: PCA of the router metrics with both a legitimate *iperf3* flow and an unaltered *picoquic* flow passing through the LL queue (router rate is set to 10Mbps).

not informative and simply represents the fact that the number of sent packets is almost constantly increasing which artificially creates a variable with large variance over time.

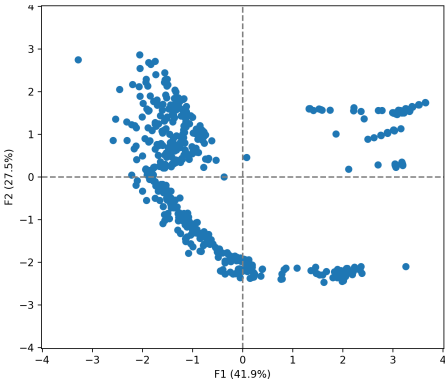
Figure 8c, showing the projection of individual measurements onto the first two principal components, clearly highlights two different behaviors separated by the occupancy of the queue: the two sets of measurements correspond respectively to an empty queue separated from moments where there is some packet lying the queue. Figure 8d shows that the measurements with empty and full queue represented by F2 are rather evenly spread regarding their occurrence in time, or the number of packets sent. Overall, besides the queue occupancy, the samples are rather evenly distributed showing no strong behavior of L4S with normal flows.



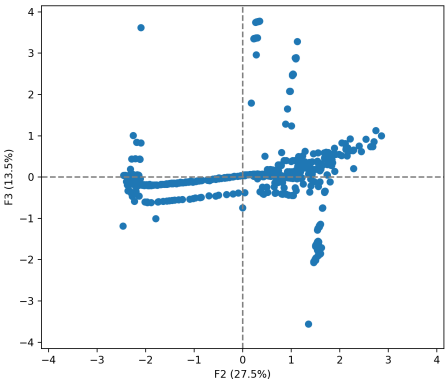
(a) Correlation circle for F1 and F2



(b) Correlation circle for F2 and F3



(c) Projection of measurement points in the first factorial plane (F1 and F2)



(d) Projection of measurement points on the plane directed by F2 and F3

Fig. 9: PCA of the router metrics with both a legitimate *iperf3* flow and an Unresponsive ECN *picoquic* flow passing through the LL queue (router rate is set to 10Mbps).

6.2.2 Unresponsive ECN

To explain the unresponsive ECN scenario depicted in Figure 9, three principal components are also required to explain 80% of the information. We can understand the first component F1 as a variable following the level of stability, as it decreases when the low latency delay and the marking increases. Figure 9a represents F1 which explains 41.9% of the data and is related to marking, low latency queue delay and sent data.

The second component F2, in return, is more linked to congestion, following the marking probability and the amount of low-latency packets in the dedicated queue and the number of packet drops. We deduce this interpretation of F2 because it is mainly made of the amount of low-latency packets, the marking probability and also the number of dropped packets. The third component F3

in Figure 9b is related to the number of bytes sent. As for the legitimate case, F3 is only showing a constantly evolving variable which cannot be represented by its mean and expectation.

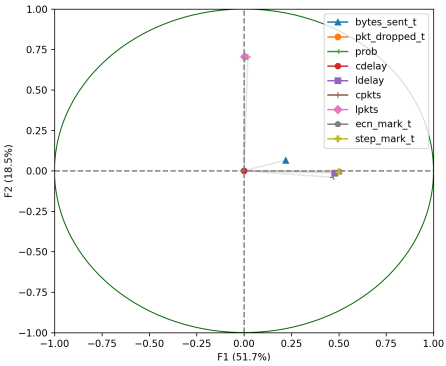
Figure 9c shows measurements along such a stability, along the horizontal axis, and the congestion level, along the vertical axis. In the bottom right corner, we can see the initial phase of the attack, stable with no congestion. Then the stability decreases while the congestion level grows. This phenomenon is visible in the bottom left corner. In the top right corner we can observe the stability of the moment where all the bandwidth has been stolen. The top left corner is the unstable phase where the congestion level keeps growing. Figure 9d shows the same distinction of measurement points along the third component, like that of the legitimate situation (i.e. with waiting times and sending time, across the growing of the congestion level).

This reveals here that a detector of the unresponsive ECN attack should monitor the progressive stability decrease in relation with a global congestion build up. Key indicators to that aim are the level of marking and the progressive increase of the amount of low-latency packets and of the marking probability. As it is a progressive phenomenon, a statistical analysis of measurements individually can hardly allow the detection of such a misbehaving pattern.

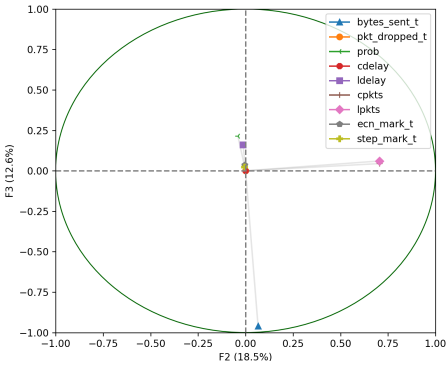
6.2.3 Application Layer Bursts

Figure 10 presents the case of the misbehaving flow, which still needs three principal components to reach 80% of comprehension. Figure 10a shows the first one which explains 51.7% of the data, and is composed of the amount of ECN and step marks, the low latency queue delay and the marking probability. The second component F2 (18.5% of data information) is linked with queue occupancy (both queues), and on Figure 10b, we can see that the third component (12.6%) is clearly related to the number of bytes sent and in relation with the marking probability elevation. Interestingly, one can note that those three main principal components are strikingly closed to the one obtained in the case of legitimate traffic only hence the analysis and understanding of those principal components remain the same: one can understand the first component as the exceeding of the step threshold, which is an event that happens when a burst occurs leading to a queue-building behavior in the LQ. The second one tends to represent that queue building behavior and the third accounts for the ever-evolving number of packet sent.

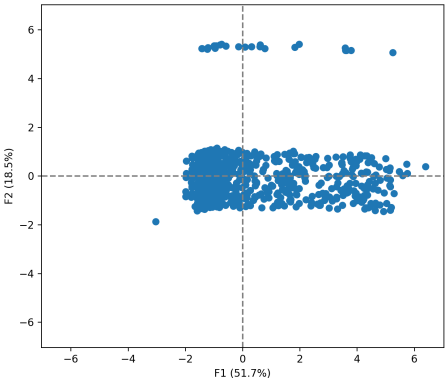
Figure 10c shows the projection of the distribution of measurement points according to the step threshold overshoot on the horizontal axis and to the queue building behavior on the vertical one. We can observe that most of the points are concentrated at the left of the cloud. This can be explained by the fact that most of the time, the threshold is respected. We can see that along the vertical axis, the cloud is quite homogeneous, except for few points that heavily represent moments when a queue building behavior occurs, corresponding to spikes in the queue occupancy. In Figure 10d we can see that the starvation



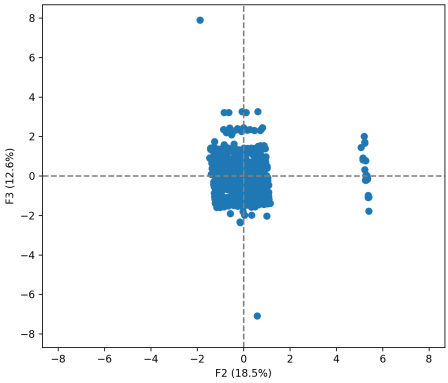
(a) Correlation circle for F1 and F2



(b) Correlation circle for F2 and F3



(c) Projection of measurement points in the first factorial plane (F1 and F2)



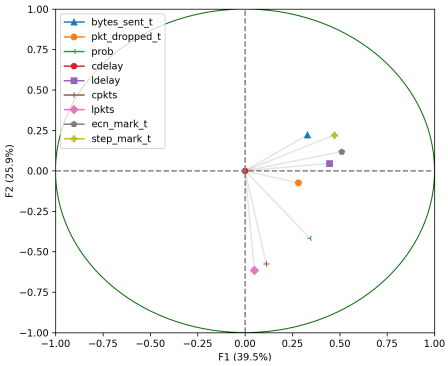
(d) Projection of measurement points on the plane directed by F2 and F3

Fig. 10: PCA of the router metrics with both a legitimate *iperf3* flow and an bursty *picoquic* flow passing through the LL queue (router rate is set to 10Mbps).

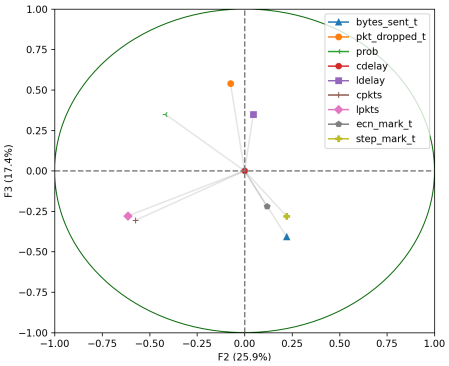
level is regular. There is a concentration of measurement points on the vertical center, corresponding to moments when no burst happens. Due to the erratic oscillating pattern inherent to bursts, we observe an equilibrium of our data around those two axes, except for spikes of component two.

To conclude about some monitoring insights, it is striking that burst traffic behaves very much alike the legitimate one and hence is expected to be rather difficult to detect. In addition, if one wants to prevent bursts within the low-latency queue, we can compare the number of step marks, if it is close to the total amount of ECN mark, this means that something is happening in the low latency queue, as the threshold is overshot.Then, one can look at the rate utilization, as bursty traffic seems to make the link under-utilized. Key indicators to that aim are the amount of ECN marked packets and step marks and the evolution of the number of bytes sent by the router.

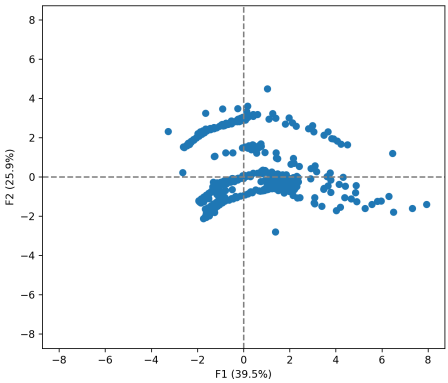
6.2.4 Unpaced Transport Flow



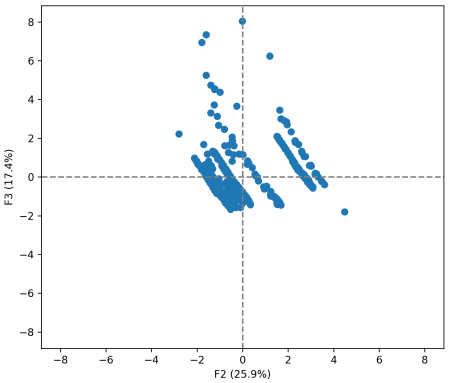
(a) Correlation circle for F1 and F2



(b) Correlation circle for F2 and F3



(c) Projection of measurement points in the first factorial plane (F1 and F2)



(d) Projection of measurement points on the plane directed by F2 and F3

Fig. 11: PCA of the router metrics with both a legitimate *iperf3* flow and a malformed (unpaced) *picoquic* flow passing through the LL queue (router rate is set to 10Mbps).

For the last situation of unpaced flow, we also need three principal components to be able to explain 80% of the data. We can understand the first component F1 as the saturation of the low latency queue, as this component is linked to dropped packets and to every aspect of the LL congestion management. Indeed F1 (39.5%) is composed of the amount of marking (both types), the delay from the LL queue, the marking probability and the number of bytes sent and dropped packets, as shown in Figure 11a. By contrast, the second component F2 can be interpreted as the transmission opportunity, or the level of low-latency-friendliness of the state of the router, which is threatened when the queue occupancy and the marking probability increases. F2 (25.9%) is

bound to the queue occupancy (a clear relation with both queues), step marks and related with marking probability.

The third one represents the starvation of the sending interface due to too many packet drops while having a large transmission opportunity. Surprisingly, the number of marked packets (either regular ECN marks or step marks) is related to the marking probability along with the number of dropped packets. This is probably due to the fact that we observe the same phenomenon as for the misbehaving traffic in Figure 10. The starvation is due to endpoint's reaction to congestion and results in an under-utilization of the link, so the router has a high-marking probability while having nothing to send for short periods of time. In Figure 11b, we can see that F3 (17.4%) is influenced by the amount of packet drops, the marking probability evolution, the LL queuing delay and it is related to the number of bytes sent, the queue occupancy and the amount of marking.

Figure 11c shows different levels of low-latency friendliness. We can see that the fewer the saturation is, the better the low-latency friendliness is. We can understand that those distinct levels are corresponding to the stationary situation of the microbursts, in term of occupancy of the value of a metric. Figure 11d shows the starvation for different levels of transmission opportunities. Most of the points are not starving as we can see in the down-left corner.

To detect such traffic, we can use the information about link-utilization, a deficiency in this indicator seems to clearly demonstrate moments where the router is free to send data but the participant are all recovering from a severe congestion event. To that aim, we can invoke the same key indicators as for the burst case, because it produces the same effects from the same causes, the difference lies in the time-scale and power of the impacts. The case of unresponsive ECN should be handled differently, but for bursts or microbursts, the detection can be done by monitoring a decrease in the link utilization, but this imply letting a first congestion event occur. To prevent that, we can monitor the component two which quantify the low-latency-friendliness level of the traffic. This can be done by watching the queue occupancy of each queue.

7 Conclusion

The L4S architecture is a promising approach to deliver low-latency content under a few milliseconds. But, to be deployed in operational networks, such a solution should be robust against attacks and any form of undesirable yet legitimate flows. In this paper, we have exposed to what extent the L4S architecture can be threatened by several types of undesirable flows. By implementing and evaluating three types of abnormal flow behaviors, we have quantified their impact and demonstrated that (1) the current L4S architecture cannot efficiently deal with them in all situations, and (2) the low-latency requirement can be eventually defeated as well as the jitter and the sending rate. We extended these experiments to different levels of throughput and for different ECN configurations in order to extend the understanding of undesirable flow interactions with L4S reactions.

The three main threats have been evaluated independently to isolate their effect on L4S, and three experiments were statistically analysed through a Principal Component Analysis, which let us access a better understanding of phenomena related to their operational impacts. This paper fulfils its intended goal of characterizing impacts of undesirable flows on a low latency flow, and PCA gives some clues and incentives to pursue research in threat detection. We showed that key indicators can be monitored to detect some undesirable flows. Besides, as for any experimental study, some phenomena would require more investigation to carefully understand their origin and potential impact. This is for instance the case of the *picoquic* traffic source for which we experienced some throughput limits and on/off patterns that would require a dedicated fine-grained investigation.

Given this assessment of L4S threats, research directions we recommend for defense and countermeasures considerations are as follows. First of all, the network community needs to put efforts on fine-grained monitoring in order to detect short time-scale phenomena like micro-bursts. We were not able to perform a per-packet analysis of micro-bursts due to the time granularity of the measurements, thus we were only able to see them indirectly as an aggregation of packets and their consequences on other metrics and on the overall performance. For mitigation from the network, we may consider adopting a per-flow scheduling strategy to confine microbursts or application-layer bursts within their own queue. Another interesting direction, as indicated in Pronto project [35], is to monitor the individual flow contribution to the shared buffer in order to detect non-fairness of the flow occupancy. A review on low latency threat detection techniques also considers some work involving Long Short-Term Memory (LSTM) and other IA-oriented solutions. Yet, further studies need to be made to be able to properly evaluate to what extent those solutions inject additional delay in the end-to-end latency. This is what our ongoing work is about. It concerns undesirable flows detection, thanks to the insights revealed by the PCA. We especially plan to investigate the design of detection algorithms adapted to the context of low latency networks which actually stands for a relevant research challenge. Indeed, in order to detect such attacks, via for example traffic pattern analysis (e.g. inter-arrival time) or via machine learning techniques, the challenge will be to find a balance between detection accuracy and detection delay penalty which may not be compatible with the LL feature, thus motivating our choice for a lightweight statistical approach. At the end, our objectives consist in (1) integrating such a detection solution in a micro-service environment and (2) to define suitable countermeasures, which can help mitigate such attacks, and consequently enable a safe and stable operation of low latency forwarding in the Future Internet.

Acknowledgments. This work is partially funded by the French ANR MOSAICO project, No ANR-19-CE25-0012.

Declarations

Some journals require declarations to be submitted in a standardised format. Please check the Instructions for Authors of the journal to which you are submitting to see if you need to complete this section. If yes, your manuscript must contain the following sections under the heading 'Declarations':

- Funding: French ANR MOSAICO project, No ANR-19-CE25-0012.
- Conflict of interest/Competing interests: None
- Ethics approval: Yes
- Consent to participate: Yes
- Consent for publication: Yes
- Availability of data and materials: Will be provided on the project's website
- Code availability: Will be provided on the project's website
- Authors' contributions: All authors except B.M. reviewed the manuscript. M.L. wrote initial versions of every parts except 6.1, prepared every figures and collected every results, configured the testbed and developed every scripts and codes. G.D. has completed section 1, 3 and 4, prepared table 2, wrote section 6.1. R.C. completed part 6 on PCA analysis and produced the initial script for PCA analysis. B.M. gave his expertise in L4S and networking configuration. He contributed to section 1 and 2.

References

[1] Albisser, O. and De Schepper, K. and Briscoe, B. and Tilmans, O. and H. Steen, DUALPI2 - Low Latency, Low Loss and Scalable (L4S) AQM, March 2019, NetDev 0x13, Prague, Czech Republic, EU

[2] B. Briscoe and K. De Schepper and M. Bagnulo and G. White, *Low Latency, Low Loss, Scalable Throughput (L4S) Internet Service: Architecture.* draft-ietf-tsvwg-l4s-arch-10, 2021.

[3] Bob Briscoe, Mirja Kühlewind, Richard Scheffenegger, *More Accurate ECN Feedback in TCP.* draft-ietf-tcpm-accurate-ecn-15, 2021

[4] D. B. Oljira and K. J. Grinnemo and A. Brunstrom and J. Taheri, *Validating the Sharing Behavior and Latency Characteristics of the L4S Architecture.* SIGCOMM Comput. Commun. Rev., vol. 50, pp.37-44, 2020.

[5] B. Mathieu and S. Tuffin, Evaluating the L4S Architecture in Cellular Networks with a Programmable Switch, 26th Symposium on Computers and Communications, ISCC 2021, Athens, Greece, September 5-8, 2021

[6] A. Hutchings and R. Clayton, Exploring the Provision of Online Booter Services, Deviant Behavior 37 (10), pp 1163-1178, Routledge 2016

[7] M. Letourneau, K. B. N'Djore, G. Doyen, B. Mathieu, R. Cogranne and H. N. Nguyen, "Assessing the Threats Targeting Low Latency Traffic: the Case of L4S," 2021 17th International Conference on Network and Service Management (CNSM), 2021, pp. 544-550, doi: 10.23919/CNSM52442.2021.9615534.

[8] B. Briscoe, K. De Schepper, O. Tilmans, M. Kühlewind, J. Misund, O. Albisser and A. Sajjad Ahmed, Implementing the 'Prague Requirements' for Low Latency Low Loss Scalable Throughput (L4S), Netdev 0x13, 2019

[9] S. Floyd and K. K. Ramakrishnan and D. L. .Black, *The Addition of Explicit Congestion Notification (ECN) to IP.* RFC3168, 2001.

[10] K. De Schepper and B. Briscoe and G. White, *DualQ Coupled AQMs for Low Latency, Low Loss and Scalable Throughput (L4S).* Internet Engineering Task Force, 2021.

[11] Koen De Schepper, Olga Bondarenko, Ing Jyh Tsang and Bob Briscoe, PI^2: A Linearized AQM for both Classic and Scalable TCP, CoNEXT, pp 105–119, 2016

[12] Rohit P. Tahiliani and Hitesh Tewari, Implementation of PI^2 queuing discipline for classic TCP traffic in ns-3, Networking, pp 1–6, IEEE Computer Society, 2017

[13] Zargar, Saman Taghavi, James Joshi, and David Tipper. "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks." IEEE communications surveys & tutorials 15.4 (2013): 2046-2069.

[14] A. Nasrallah et al., "Ultra-Low Latency (ULL) Networks: The IEEE TSN and IETF DetNet Standards and Related 5G ULL Research," in IEEE Communications Surveys & Tutorials, vol. 21, no. 1, pp. 88-145, Firstquarter 2019, doi: 10.1109/COMST.2018.2869350.

[15] Ergenç, D., Brülhart, C., Neumann, J., Krüger, L., & Fischer, M. (2021, June). On the security of IEEE 802.1 time-sensitive networking. In 2021 IEEE International Conference on Communications Workshops (ICC Workshops) (pp. 1-6). IEEE.

[16] E. Grossman, T. Mizrahi and A. Hacker. Deterministic Networking (DetNet) Security Considerations. RFC 9055. IETF June 2021.

[17] T. Yoshizawa, S. B. M. Baskaran and A. Kunz, "Overview of 5G URLLC System and Security Aspects in 3GPP," 2019 IEEE Conference on Standards for Communications and Networking (CSCN), 2019, pp. 1-5, doi: 10.1109/CSCN.2019.8931376.

[18] M. A. Javed and S. khan Niazi, "5G Security Artifacts (DoS / DDoS and Authentication)," 2019 International Conference on Communication Technologies (ComTech), 2019, pp. 127-133, doi: 10.1109/COMTECH.2019.8737800.

[19] N. Kothari and R. Mahajan and T. Millstein and R. Govindan and M. Musuvathi, *Finding Protocol Manipulation Attacks.* SIGCOMM Comput. Commun. Rev. 41(4):26-37, 2011.

[20] R. Sherwood and B. Bhattacharjee and R. Braud, *Misbehaving TCP Receivers Can Cause Internet-Wide Congestion Collapse.* ACM Conference on Computer and Communications Security, pp.383-392, 2005.

[21] A. Laraba and J. François and S. R. Chowdhury and I. Chrisment and R. Boutaba, *Mitigating TCP Protocol Misuse With Programmable Data Planes.* IEEE Transactions on Network and Service Management, vol. 18, no. 1, pp.760-774, 2021.

[22] D. Ely and N. Spring and D. Wetherall and S. Savage and T. Anderson, *Robust congestion signaling.* International Conference on Network Protocols. ICNP 2001, pp.332-341, 2001.

[23] A. Laraba and J. François and I. Chrisment and S. R. Chowdhury and R. Boutaba, *Defeating Protocol Abuse with P4: Application to Explicit Congestion Notification.* IFIP Networking Conference, pp.431-439, 2020.

[24] W. Zhijun and L. Wenjing and L. Liang and Y. Meng, *Low-Rate DoS Attacks, Detection, Defense, and Challenges: A Survey.* IEEE Access, vol. 8, pp.43920-43943, 2020.

[25] Bianchin, G., Pasqualetti, F. (2018). Time-Delay Attacks in Network Systems. In: Koç, Ç.K. (eds) Cyber-Physical Systems Security. Springer, Cham. 2018. https://doi.org/10.1007/978-3-319-98935-8_8

[26] C. Chen, Y. Chen, K. Zhang, M. Ni, S. Wang and R. Liang, "System Redundancy Enhancement of Secondary Frequency Control Under Latency Attacks," in IEEE Transactions on Smart Grid, vol. 12, no. 1, pp. 647-658, Jan. 2021, doi: 10.1109/TSG.2020.3012977.

[27] K. S. Xiahou, Y. Liu and Q. H. Wu, "Robust Load Frequency Control of Power Systems Against Random Time-Delay Attacks," in IEEE Transactions on Smart Grid, vol. 12, no. 1, pp. 909-911, Jan. 2021, doi: 10.1109/TSG.2020.3018635.

[28] Aggarwal, P., Gonzalez, C., Dutt, V. (2016). Cyber-Security: Role of Deception in Cyber-Attack Detection. In: Nicholson, D. (eds) Advances in Human Factors in Cybersecurity. Advances in Intelligent Systems and

Computing, vol 501. Springer, Cham. 2016 https://doi.org/10.1007/978-3-319-41932-9_8

[29] Q. Zhang, K. Liu, Y. Xia and A. Ma, "Optimal Stealthy Deception Attack Against Cyber-Physical Systems," in IEEE Transactions on Cybernetics, vol. 50, no. 9, pp. 3963-3972, Sept. 2020, doi: 10.1109/TCYB.2019.2912622.

[30] Xiaohua Ge, Qing-Long Han, Maiying Zhong, Xian-Ming Zhang, Distributed Krein space-based attack detection over sensor networks under deception attacks, Automatica, Volume 109, 2019, 108557, ISSN 0005-1098, https://doi.org/10.1016/j.automatica.2019.108557.

[31] Wang, Kunyu, *et al.* "Resilient control of networked control systems under deception attacks: a memory-event-triggered communication scheme." International Journal of Robust and Nonlinear Control 30.4 (2020): 1534-1548.

[32] Addanki and L. Iannone, "Moving a step forward in the quest for Deterministic Networks (DetNet)," 2020 IFIP Networking Conference (Networking), 2020, pp. 458-466.

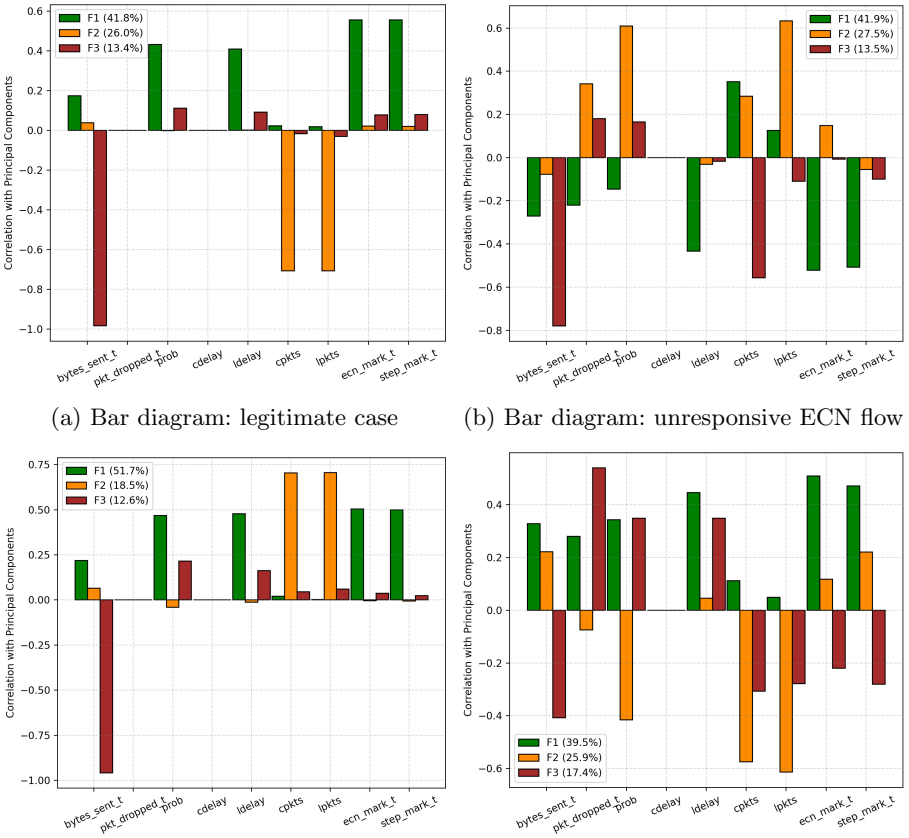
[33] Henrik Steen, *Destruction Testing: Ultra-Low Delay using Dual Queue Coupled Active Queue Management.* Masters Thesis, Dept of Informatics, Uni Oslo, 2017

[34] J. Iyengar and M. Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. Internet-Draft (draft-ietf-quic-transport-17) available at https://datatracker.ietf.org/doc/html/draft-ietf-quic-transport-17. IETF 2019.

[35] Nick McKeown et al., *The network as a programmable platform: fertile new ground for networking research.* SIGCOMM Technical session, 2020.

Appendix A Additionnal infos on PCA

To complete the comprehension of PCA, we provide bar diagrams indicating to what extent each metrics is contributing to the two first components. Each principal component is a linear combination of the metrics whose weighted coefficient are depicted in ordinate.



(a) Bar diagram: legitimate case

(b) Bar diagram: unresponsive ECN flow

(c) Bar diagram: low latency burst case

(d) Bar diagram: malformed flow

Fig. A1: PCA of the router metrics with both a legitimate *iperf3* flow and an malformed (unpaced) *picoquic* flow passing through the LL queue (router rate is set to 10Mbps). Bar diragramm representations