Assessing the Threats Targeting Low Latency Traffic: the Case of L4S

Marius Letourneau*, Kouame Boris N'Djore*, Guillaume Doyen†, Bertrand Mathieu‡, Rémi Cogranne*
Huu Nghia Nguyen§
*LIST3N, University of Technology of Troyes, Troyes, France, {first.last}@utt.fr
†OCIF - IRISA (UMR CNRS 6074), IMT Atlantique, Rennes, France, guillaume.doyen@imt-atlantique.fr
‡Orange Innovation, Lannion, France, bertrand2.mathieu@orange.com
§Montimage, Paris, France, huunghia.nguyen@montimage.com

***Abstract*—New types of services with low-latency requirements have become a major challenge for the future Internet. Many optimizations, all targeting the latency reduction have been proposed. Among them, jointly re-architecting congestion control and active queue management has been particularly considered. In this effort, the L4S (Low Latency, Low Loss and Scalable Throughput) proposal aims at allowing both classic and low-latency 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 low-latency feature and consequently make their supported applications unusable. In this paper, we analyze a set of weaknesses of L4S architecture and 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 undesirable flows in a real testbed and evaluating how they impact the proper delivery of low-latency flows, we demonstrate their reality and relevance for future deployments.**

I. Introduction

Years after years, 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 type of services. The Low Latency, Low Loss and Scalable Throughput (L4S) architecture is currently being discussed in the IETF [1] and acts as a promising candidate solution to ensure these low network latency requirements.

However, if such novel architectures exhibit satisfying performance under normal traffic conditions, the question of their capability to deal with abnormal traffic is still an open issue. For instance, the ability of L4S to satisfy LL requirements while maintaining a well balance with classic traffic makes it highly sensitive to non-regular traffic, as illustrated in [10] who studied the impact of traffic bursts 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 booters [17], 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 this paper, we identify the main attacks a malicious user can implement against L4S and present their impacts. To that aim, we have implemented a testbed hosting L4S, in which we have generated some LL and classic flows exhibiting a legitimate or undesirable behavior. The latter is achieved by hacking the behavior of QUIC which implements its congestion control within the application layer, thus making it easily accessible to any malicious user. Our evaluation proved that these attacks can induce several issues for the legitimate traffic such as unfair bandwidth sharing, increased queuing delay or highly unstable throughput. This largely degrades quality and eventually makes the LL applications not reaching their low-latency constraints.

The rest of the paper is structured as follows. Section II presents the background, mainly the L4S architecture and the work related to undesirable flow generation for malicious usage. Section III presents the experimental setup implemented to perform our measurements. Section IV details and explains the results collected with three different attack patterns, before we conclude this paper in Section V.

II. Background and related work

A. The L4S Architecture

The L4S architecture is currently under standardization at the IETF [1] and focuses on reducing queuing delay for flows with a low-latency requirement. Coexistence and fairness between low-latency flows and classic flows are strong prerequisites in the design of L4S. This is achieved leveraging a scalable congestion control such as Prague [16], Explicit Congestion Notification (ECN) [2] and a Dual Queue Coupled Active Queue Management (DQC AQM) [3] [13].

From the endpoint side, congestion control and the network stack is adapted to improve scalability and RTT-independance. Prague is known as such a congestion control (a.k.a. scalable congestion control) and is available with TCP [16] or with some QUIC implementations. The main idea is to adapt the reduction of the congestion windows to the actual level of

978-3-903176-36-2 ©2021 IFIP

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 [12].

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 L4S flows by checking the ECN flags of the IP header. The second component is a coupling mechanism (e.g., [13]) and the last one is a scheduler that may absorb small packet bursts. The coupling mechanism, which is the core of the L4S proposal, works as follow. For the classic queue, a PI² AQM is proposed [14] and is easy to implement [15]. A PI[1] controller generates a packet marking probability based on 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 equation:

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

where τ_0 is the target value, T is the period used to recompute the probability, set by default to 16ms, α is a weight associated with the error regarding the target and β is a weight 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 explained previously multiplied by a coupling factor k (by default set to 2). 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 AQM, 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 which ensures that low-latency flows do not starve classic flows due to over-marking from the router and over-reaction from the classic flow.

B. Related work

Given the L4S architecture described above, we subsequently review the set of works which focus on undesirable flow generation which may impact the good operation of L4S. We use the term *undesirable flows* when referring to unresponsive flows, malformed flows and misbehaving flows, which includes both legitimate and attack flows.

1) Misbehaving Flows: Misbehaving flows can be classified in two categories: protocol manipulation and low-rate DoS.

A protocol manipulation is the ability of some of the participants to subvert the protocol without the knowledge of the others [4]. Most of these attacks are TCP-centered. We can first mention acknowledgement manipulation attacks, also called *hacked ACK* which manipulate a TCP endpoint to make the victim saturates the network (more specifically an edge router shared by the targeted victim). The optimistic acknowledgment (*opt-ack*) attack [4], [5], [8] 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 manipulate ECN, we call it *hacked ECN*. One can conceal the congestion notification by not informing the remote entity of the communication [6], [4], [7]. An attacker can also generate false congestion notification in order to steal more bandwidth. Another hacked ECN situation is the case where a malicious user acts in a protocol-compliant way but, when a congestion occurs, he/she sends the correct signalling to inform that he/she has reduced his rate while not doing so. This attack generates an unresponsive ECN-capable traffic and subsequently, we term it *Unresponsive ECN*. Some research for mitigating protocol manipulation has been made recently in [8] by designing 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 focused on detecting flows that are not protocol-compliant. However the case of a misbehaving but protocol-compliant flow has not be covered yet to our knowledge and might be more subtle to differentiate from a legitimate flow.

The other category of misbehaving flows concerns low-rate DoS attacks (LDoS) whose general model is described in [9]. Its principle is to send 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 regular 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.

2) Unresponsive flows: An unresponsive flow is a flow that does not respond to congestion signaling (ECN marks, dropped packets or additional delay). It can be legitimate UDP or VoIP traffic or any protocol that does not implement a congestion control. It can be introduced in any of the classic or L4S queue and can lead to overloading queues or to congestion signal saturation. As mentioned by the IETF in [3], 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.

3) Malformed flows: Malformed flows are usually legitimate but undesirable from a L4S perspective. For instance, a bursty behavior may occur when the network stack of regular

[1]Proportional Integral

operating systems within endpoints waits for the sending buffer to be filled before sending data over the network. This results in an on/off pattern that injures the L4S performance. DualPI² overloading has been studied in [11]. 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.

In [10], 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) and to the burstiness of the L4S architecture itself. The authors tested TCP Prague on DualPI² and showed that it performs a better sharing-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.

4) Countermeasures: The IETF has identified the above mentioned undesirable flows as an issue [1], [3] and it proposes some basic countermeasures to handle unresponsive flows by sacrificing some performances (L4S delay, L4S throughput or introducing L4S drop). Traffic shaping and traffic policing (or queue protection) are also considered for malformed flows. However, classical techniques for traffic shaping are not straightly applicable in a LL context, as they may lead 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's network scheduler, it can drastically reduce traffic (micro-)burstiness. Besides, a solution has also been proposed by the IETF to some protocol manipulation, and more specifically for the case of ECN concealing. 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. However, if all these issues related to undesirable flows have been identified by the IETF, to the best of our knowledge, there is neither any comprehensive study of their impact on LL traffic, which is the purpose of this paper, nor any implementation of dedicated detection and mitigation components such as those cited above, which is the purpose of our research project.

III. Experimental Setup

In order to understand and comprehensively evaluate how the L4S architecture behaves when some of the identified undesirable flows happen and what the possible impacts on a legitimate LL flow are, we set up a testbed and implemented the three aforementioned categories of undesirable flows.

A. Testbed

As depicted in Figure 1, we use the following topology. A baremetal server act as a router and is equipped with an Intel(R) Xeon(R) CPU E5-2430 v2 @ 2.50GHz, and two Intel Corporation Ethernet 10G 2P X520 Adapter runs DualPI². One Virtual Machine for each endpoint is hosted in an OpenStack cloud platform. L4S nodes are using the Linux image from the L4STeam[2] 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 different interfaces. On the egress direction of *eno1* (receivers side), DualPI² is configured with a 10 Mbps rate, all other parameters 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 15ms 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 DualPI² will always mark exceeding packets within the low latency queue, is set to 1ms.

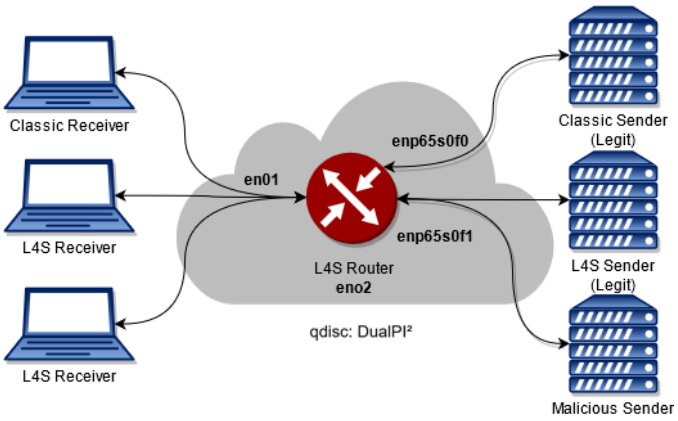

Fig. 1: Network topology of our testbed

The legitimate flows are generated with $iperf3$ and controlled by TCP Prague while the malicious node is using QUIC, a protocol based on UDP which is enhanced in the userspace with congestion control algorithms and other features that are present in TCP. This lets a malicious user to easily modify the sending behavior without having to deal with the Linux kernel networking stack. We especially chose to use $picoquic$[3], a minimalist implementation of QUIC which has been forked by the L4S creators to support the Prague congestion control. We can adjust $maxrate$ and enable or disable pacing with the system program traffic control (tc).

The base RTT is set to 15ms with $wondershaper$ in server's interface. Metrics are collected from the endpoints with calls to socket statistics (ss) 4 times per base RTT. This provides the tcp_info data structure of the kernel, which contains reported RTT and its mean deviation, the congestion window ($cwnd$), maximum potential sending rate based on the formula:

[2]https://github.com/L4STeam/linux
[3]https://github.com/private-octopus/picoquic

$\frac{cwnd \times MSS}{RTT}$ and last bytes acked. For the router, metrics are collected at the same frequency using tc, which provides metrics from DualPI² such as classic and low latency queue occupation and queuing delay, marking probability from the PI controller (base probability), the amount of ECN marks and the amount of dropped packets. Finally, each experiment lasts 60 seconds.

B. Reference Traffic

In order to highlight and quantify the impact of undesirable flows, we need to have a control sample to compare with our different scenarii. To that aim, we consider that the measurements depicted in Figure 2 act as the baseline of our experiments. The reader can refer to it to identify the traffic alterations due to the different undesirable flows we consider subsequently. Figure 2 shows regular network conditions when there is one classic flow and one low latency flow. In that situation, the router sends 9.54 Mbps, shared, in accordance with Figure 2.b, which represents the sending rate. The maximum potential sending rates are depicted in blue and yellow and the red line indicates the actual data rate, based on received acked bytes. Given the behavior of TCP Prague, it tries to takes all the available bandwidth. For the classic flow, the data rate is limited to 5 Mbps to make sure that the observed effects are not due to natural saturation of the router. Figure 2.c shows the differentiated queue delay in the LL and the classic one and the corresponding RTT is represented in Figure 2.a. We can observe that the RTT of the classic flow is much more variable than for the LL flow which is close to the base RTT set to 15ms in our case. As we can see, flow congestion is well handled by ECN marking and even though the LL traffic wants to take all the possible bandwidth, Figure 2.e shows that no packet drop is necessary to guarantee the flows respective requirements. The number of marked packets is measured in Figure 2.f. For Low-Latency packets, the decision is based on the maximum between the probability of LL AQM (which corresponds eventually to the number of step marks also showed in Figure 2.f) and the base probability of the classic AQM, showed in Figure 2.d, multiplied by the coupling factor k.

IV. Results Analysis

This section explores to what extent L4S is affected by an unexpected behavior in the presence of undesirable flows.

A. Misbehaving flow: Unresponsive ECN

As a misbehaving flow, we first propose to implement an ECN abuse attack targeting the increasing of a legitimate flow latency. The protocol manipulation we implemented consists in being unresponsive to congestion notification while respecting ECN signaling. This was implemented by removing the congestion windows reduction when updating the coefficient of reduction in Prague. As such, it expects to saturate the LL queue to increase the marking probability (leading eventually to some packet drops) and thus generate an extra delay, sufficient enough, 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 from legitimate users.

Figure 3 shows time series for the different metrics we consider in the context where the malicious user is 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 is twice bigger than that of Figure 2. Unresponsive ECN puts DualPI² to saturation and triggers the $drop\_on\_overload$ reaction. The amount of ECN ma rks is twice bigger than step marks (i.e. amount of ECN marks due to exceeding the threshold). We can also notice that the marking probability is around 50% yet rather stable. The sending rate is also stable yet very low. This is due to the fact that the legitimate flow responds correctly to congestion notification while the attacker steals all the available bandwidth pretending reducing the sending rate.

B. Unresponsive flow: Bursts

For the unresponsive behavior, we generate traffic bursts from a classic sender to disturb a legitimate flow in the low latency queue. To that aim, we configure the ECN flags of the attack traffic to be classified in the classic queue. We saturate the low latency queue with traffic from the classic one. The attack consists here in playing with the coupling mechanism. The bursty behavior will trigger a high queue variation which is more likely to increase the marking probability due to the PI proportional gain β in comparison with a constant saturation.

Figure 4 shows the results we collected with such a malicious user generating a bursty flow within the classic queue. The marking probability is erratic but is around 20% in average. Both ECN marks and step marks have a step pattern and occur ten times less often than in case of Unresponsive ECN. 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, DualPI² reacts very quickly to punctual events, resulting in a sending rate reduction from the legitimate user which in the end induces heavy fluctuation and wide dispersion of sending rate possible values. RTT and low latency queue delay are also affected by 4ms when a burst occurs, even though the burst takes place in the classic queue. This is due to the coupling mechanism in DualPI² 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 1ms, the marking probability of the LL queue is governed by the base probability multiplied by the coupling factor. This is why, in the end, the sending rate of the legitimate low latency flow is affected.

C. Malformed flow: No pacing

A solution to avoid bursty emitting pattern to be sent from an endpoint is to pace packet emission over a RTT instead of sending a bulk. This operation is accomplished by the TCP stack of the Linux kernel and also as a queuing discipline

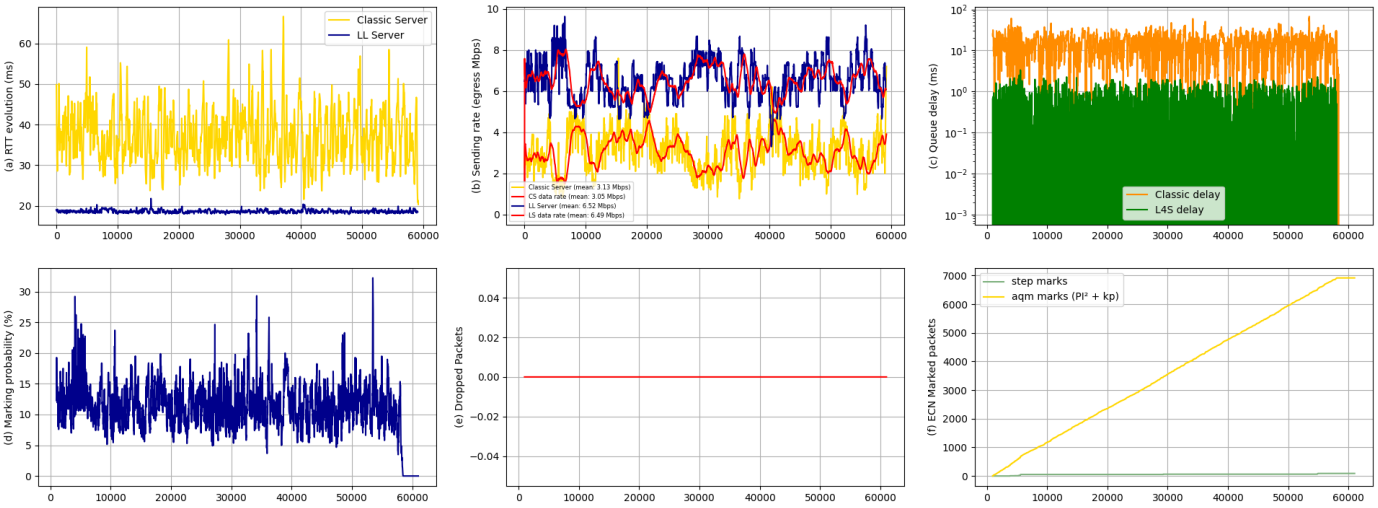

Fig. 2: Standard behavior of the L4S architecture with one LL and one classic flows. (a) RTT evolution (ms). (b) Sending rate (egress Mbps). (c) Queue delay (ms). (d) Marking probability (%). (e) Dropped packets. (f) ECN Marked packets. Horizontal axis is the time in ms.

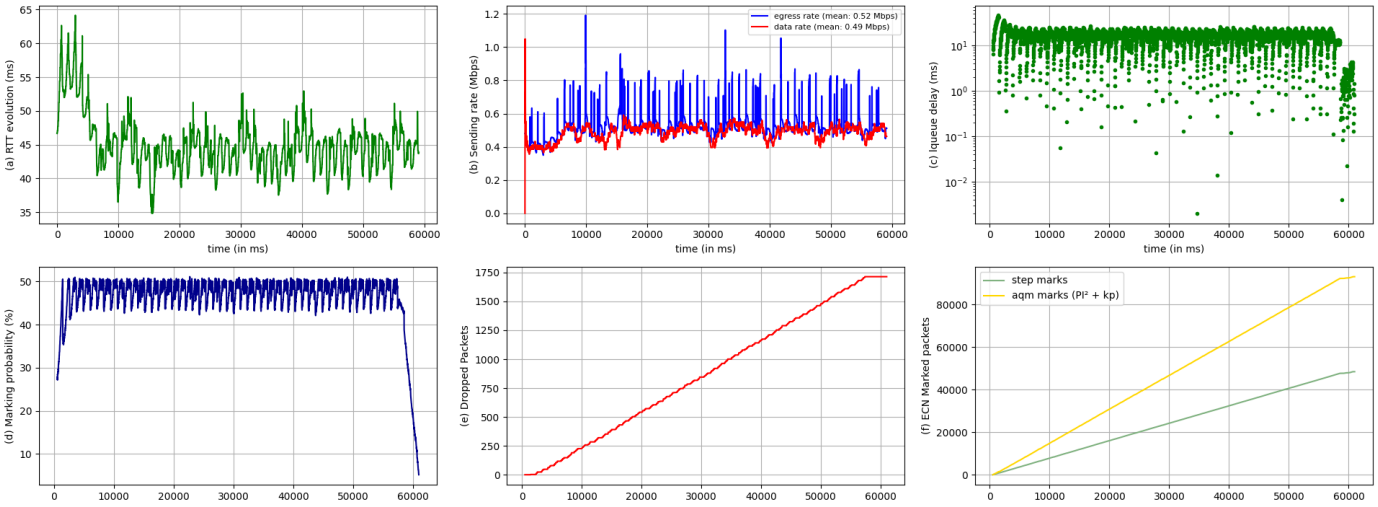

Fig. 3: Unresponsive ECN impacts on a legitimate low latency flow. The vertical axes of the subfigures are the measured metrics (two first ones show metrics from the legitimate flow, the others show metrics from the router) and the horizontal axis is the time in ms.

with Fair Queuing (FQ). Consequently, in this experiment, we used FQ as a queuing discipline at the sender endpoint, which lets us control the pacing on the egress traffic as well as the maximum rate limitation in order to control the sending rate of the application layer. When pacing is disabled, it generates undesirable flows. This kind of undesirable flow is meant to generates micro-saturation with sub-RTT bursts (also termed micro-bursts). We aim at seeing the impact of pacing and at generating micro-bursts when disabled.

Figure 5 shows the results we collected in the context where the malicious user is generating such micro-bursts. The marking probability is less erratic compared to the previous experiment, but still not stable and is in average around 50% and sometime even more, which is similar to the Unresponsive ECN case. This leads to a large number of ECN marks and by far to the highest number of dropped packets among the three experiments. RTT is very erratic, averages around 40ms, and has the widest dispersion among other measured undesirable flows of our study. We can see that it is mostly due to LL queue delay which has the widest dispersion. The sending rate is higher than for the Unresponsive ECN scenario because the legitimate flow is able to increase his congestion window between two micro-bursts.

D. Discussion

In order to clearly assess to what extent undesirable flows induce changes on the different metrics we observe as compared to our reference experiment, Figure 6 summarizes the results in boxplots.

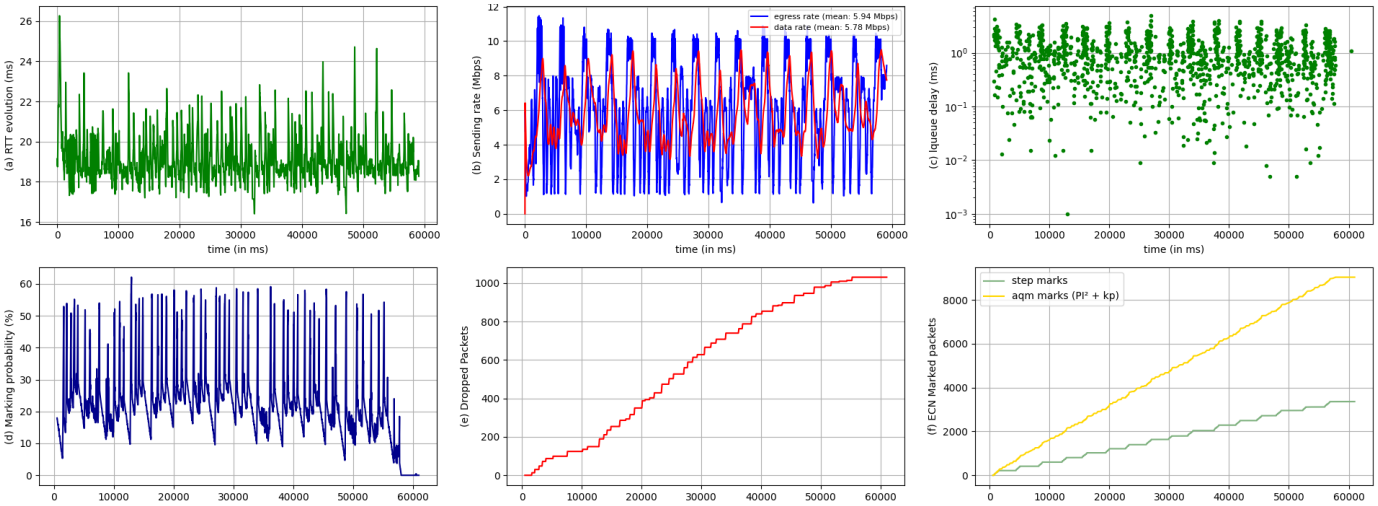

Fig. 4: Bursts within the classic queue impacts on a legitimate low latency flow. The vertical axes of the subfigures are the measured metrics (two first ones show metrics from the legitimate flow, the others show metrics from the router) and the horizontal axis is the time in ms.

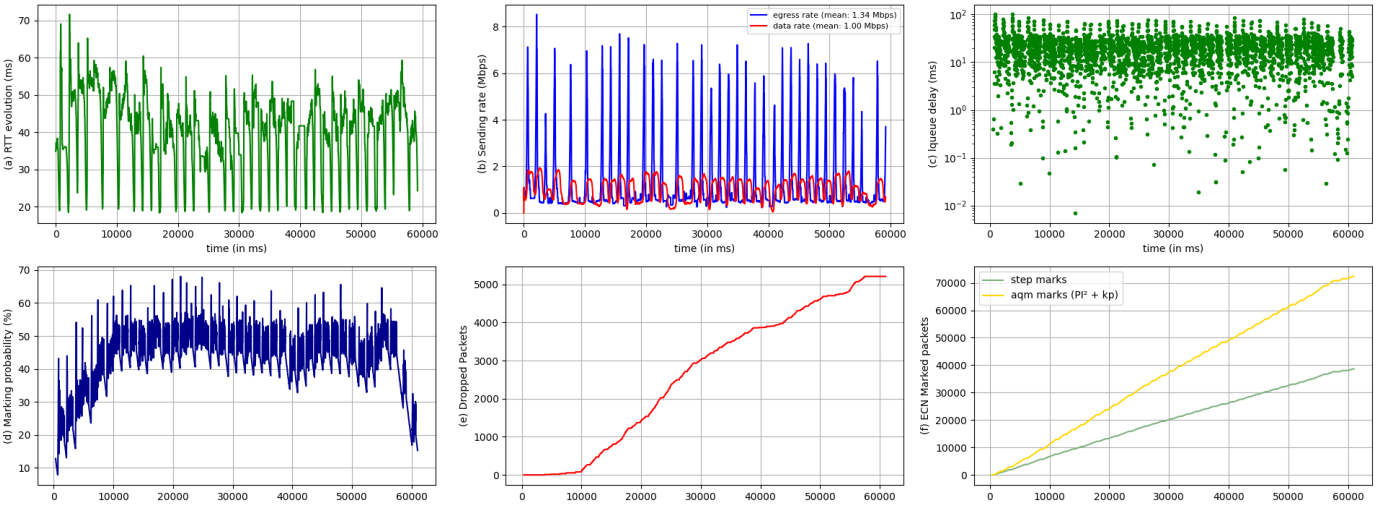

Fig. 5: Malformed flow impacts on a legitimate low latency flow. The vertical axes of the subfigures are the measured metrics (two first ones show metrics from the legitimate flow, the others show metrics from the router) and the horizontal axis is the time in ms.

Generally speaking, we can see that bursty traffic has a limited impact on the RTT when it lodge itself within the classic queue but it can bring a lot of variation in the sending rate when it passes through either queues (Figure 6.a and 6.b). Micro-bursts from malformed flows are on the other hand clearly responsible for packet drops, as we can observe in Figure 6.e. We can conclude that pacing is essential within endpoints to support the data plane performance. Unresponsive ECN impacts the marking probability, the number of marked packets and the low latency queuing delay (Figures 6.d, 6.f and 6.c). Contrasting unresponsive ECN with malformed flows, we can see that pacing results in less packet drop, thus a higher number of marked packets, a stable sending rate and RTT and limits effects on LL queue delay from the Unresponsive ECN attack. However, regardless the undesirable flow we consider, they all clearly exhibit an impact that defeats the expected property of the L4S architecture and eventually the LL applications running in endpoints.

V. Conclusion

The L4S architecture is a promising approach to deliver low-latency contents under a few milliseconds. But, to be deployed in operational networks, such a solution should be robust to attacks and to legitimate undesirable flows. In this paper, we have studied to what extent the L4S architecture can be threaten by several types of undesirable flows. By implementing and evaluating three type of abnormal flow behaviors, we have quantified their impact and proved that the current L4S architecture cannot efficiently deal with them

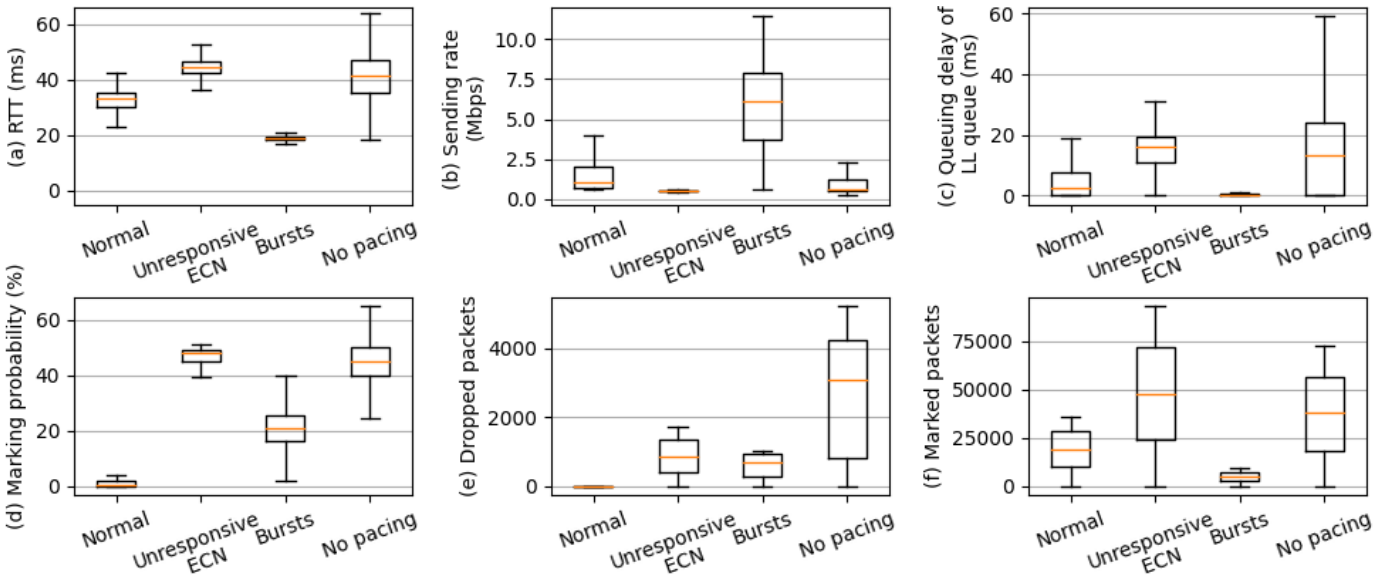

Fig. 6: Multi-criteria comparison of the impact of undesirable flows on a legitimate LL flow. Vertical axes are measured metrics represented as a boxplot with the median value and first and third quartile and the first and ninth decile. Horizontal axis is the type of undesirable flow.

since eventually the low-latency requirement is defeated as well as the jitter and the sending rate.

The three main threats have been evaluated independently to isolate their effect on L4S, but further studies need to be made to see what happen when these undesirable flows are combined. Consequently, our ongoing work concerns undesirable flows combined altogether and preliminary results seems to show that pacing at senders can drastically reduce most of the impacts. Besides, we also plan to study the impacts of each presented attacks when we vary their parameters to determine where each attack is the most powerful and by contrast how an attack can dissimulate its behavior to be undetected.

In our mid-term future work, we will investigate detection modules, which can detect such attacks, via for example traffic pattern analysis (e.g. inter-arrival time or packet size) or via machine learning techniques. Finally, our ultimate objective consists in defining countermeasures, which can help to mitigate such attacks, and consequently enable a safe and stable operation of low latency forwarding in the Future Internet.

Acknowledgment

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

References

[1] 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.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

[13] 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

[14] 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

[15] 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

[16] 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

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