Accepted for presentation in: IEEE 102th Vehicular Technology Conference (VTC-Fall), Chengdu, China, October 2025.

The NODROP Patch: Hardening Secure Networking for Real-time Teleoperation by Preventing Packet Drops in the Linux TUN Driver

Tim Gebauer, Simon Schippers, and Christian Wietfeld
Communication Networks Institute, TU Dortmund University, 44227 Dortmund, Germany
e-mail: {Tim.Gebauer, Simon.Schippers, Christian.Wietfeld}@tu-dortmund.de

***Abstract*—Throughput-critical teleoperation requires robust and low-latency communication to ensure safety and performance. Often, these kinds of applications are implemented in Linux-based operating systems and transmit over virtual private networks, which ensure encryption and ease of use by providing a dedicated tunneling interface (TUN) to user space applications. In this work, we identified a specific behavior in the Linux TUN driver, which results in significant performance degradation due to the sender stack silently dropping packets. This design issue drastically impacts real-time video streaming, inducing up to 29 % packet loss with noticeable video artifacts when the internal queue of the TUN driver is reduced to 25 packets to minimize latency. Furthermore, a small queue length also drastically reduces the throughput of TCP traffic due to many retransmissions. Instead, with our open-source NODROP Patch, we propose generating backpressure in case of burst traffic or network congestion. The patch effectively addresses the packet-dropping behavior, hardening real-time video streaming and improving TCP throughput by 36 % in high latency scenarios.**

I. Introduction

Secure networking has gained increasing importance recently due to the growing number of cyberattacks and the general intention of digitizing many public use cases. While most applications rely on secure data transmission for privacy purposes, applications like vehicle teleoperation require it for safety reasons. Aside from real-time video streams, also critical downlink control commands require secure data transmission to keep bad actors from interfering with the vehicle's control. For securing these transmissions different approaches can be chosen. First, a transport or application layer encryption approach can be selected. In this approach, each application is responsible for encrypting and decrypting its data using protocols like Transport Layer Security (TLS). While this approach secures each application independently, preventing a system-wide breach, each must support these protocols. Furthermore, each new application has to be configured individually to ensure data security. Therefore, another established approach is using so-called Virtual Private Networks (VPNs), which provide a secure tunnel between the sender and receiver system and encrypt all traffic sent through it. So, each application does not have to ensure encryption but can rely on the VPN. Accordingly, using VPNs is the obvious choice in many cases. On Linux, most VPNs rely on the TUN driver for providing a user space network interface as seen in Fig. 1. Our previous work [1] uses the multi-link aggregation software SEAMLESS [2] to enable reliable real-time teleoperation. SEAMLESS also relies on the aforementioned TUN driver to enable general compatibility

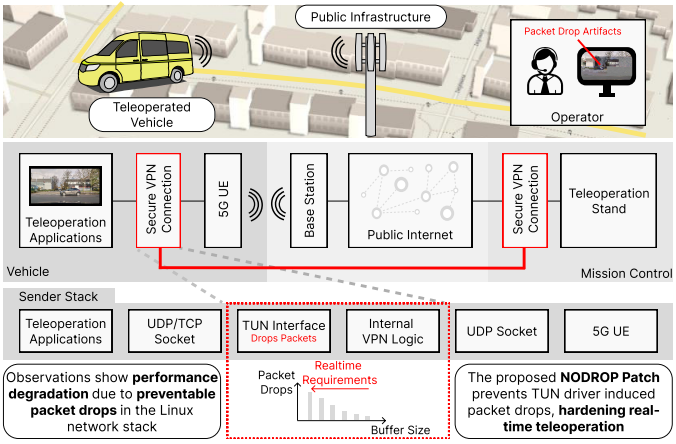

Fig. 1. Visualization of a typical VPN-enabled secure teleoperation setup, highlighting the sender stack with regard to the investigated TUN driver.

with all user space applications. To ensure minimum worst-case video latencies, we drastically reduced most of the queues on the sender side. While most queues performed as expected, we encountered unexpected packet drops when reducing the internal queue of the TUN driver below the default value, resulting in many video artifacts and unusable video streams. Our key contributions are summarized as follows:

- We identify the root cause of the packet dropping behavior within the Linux kernel TUN driver.
- Propose the NODROP Patch [3] to address these drops.
- Benchmark the patch on real-time video streaming and throughput tests over established user space VPN solutions like OpenVPN and WireGuard Go.

The remainder of this paper is structured as follows: After we discuss the related work in Sec. II, we analyze the root cause of packet dropping and propose a solution, the NODROP Patch, in Sec. III. Sec. IV focuses on an in-depth comparison of the impact on established VPN solutions. We conclude our work in Sec. V and provide an outlook for future research.

II. Related Work

Generally, real-time teleoperation has strict requirements regarding latency and throughput of the used communication link. In [4], a maximum one-way latency of 100 ms is required for a minimum data rate of 32 Mbit/s to control a teleoperated vehicle safely. In addition, [5] rather specifically specifies a round trip time requirement of a maximum of 250 ms for

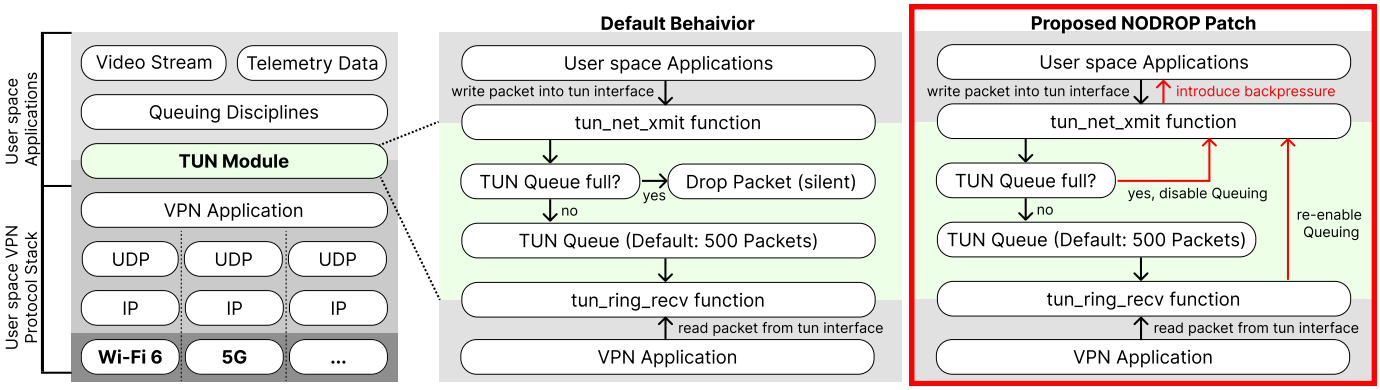

Fig. 2. Overview of the user space Linux VPN network stack highlighting the default TUN drivers behavior as well as the proposed NODROP Patch [3].

proper teleoperation. Furthermore, [6] found that the quality of teleoperation generally deteriorates at higher latencies.

The Linux kernel's TUN driver is widely used. It is a crucial part of Software Defined Radio (SDR)-based solutions like GNU radio [7], [8] and is part of open 5G solutions like e.g. openairinterface [9]. In addition, most established VPNs use it to provide user space applications with a network device for communication. Although the TUN driver, as part of the combined TUN/TAP driver [10], has been in the Linux Kernel since 1999, potential improvements are still identified [11]. Proper backpressure behavior of lower-level network layers is key for user space applications like the aforementioned adaptive video stream of [1]. Additionally, approaches like QAware Multi-Path TCP [12] and general packet scheduling approaches [13] also benefit from the backpressure.

III. Proposing the NODROP Patch

In order to identify the reason for the packet drop encountered in [1], the network statistics were first examined. This made it possible to determine that the issue was not due to transmission errors of real network interfaces but explicit TX_DROPs in the TUN interface.

1) Analysis of the default TUN driver behavior: During the in-depth analysis of the TUN driver source code, it was found that the current implementation explicitly drops packets when the internal queue overflows. This internal queue is referred to as the TUN queue in the following. Due to the packet loss, applications at a higher level of the sender stack can transmit their data as fast as possible without any feedback on potential congestion. In the case of Transmission Control Protocol (TCP), it results in many avoidable retransmissions. Furthermore, queuing disciplines (qdisc) like the 'prio' qdisc, commonly used for QoS, typically have no effect when used with a TUN interface. Fig. 2 illustrates the current implementation of the TUN driver and our proposed NODROP Patch for mitigating the packet drops, which is described in the following.

2) Implementation and Validation of the NODROP Patch: To work out a possible solution for this problem, we first compared the TUN implementation with other driver implementations of real network interfaces. It revealed that those drivers use a start/stop flow control, which disables the queuing into the interface whenever their internal queue is full. Then, after the internal queue is cleared to some extent, the queuing is reenabled. In addition to preventing packet drops, this approach allows upper-layer applications to detect congestion. Consequently, we adopted this approach for the TUN driver with the NODROP Patch. A measurement setup is implemented to support the hypothesis of the identified flaw in the TUN driver and the validity of our proposed fix. Fig. 3 illustrates the setup, which allows us to analyze the behavior and performance of the isolated TUN interface.

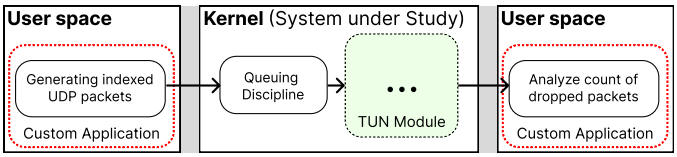

Fig. 3. Implemented validation pipeline consisting of sender and receiver with customizable packet processing rates, allowing to induce artificial backpressure.

For this purpose, we implement a custom transmission application that sends packets to the TUN interface as fast as possible. Conversely, a custom application reads packets from the TUN interface as fast as possible. With this setup, the default kernel and our proposed NODROP Patch are analyzed using different TUN queue sizes. Fig. 4 presents the result for the overall throughput and the data rate of lost traffic on an Intel© Core™ Ultra 7 165U with disabled Turbo-Boost. For the default kernel, it is shown that no packet loss is present for TUN queue sizes of 1000 packets or more.

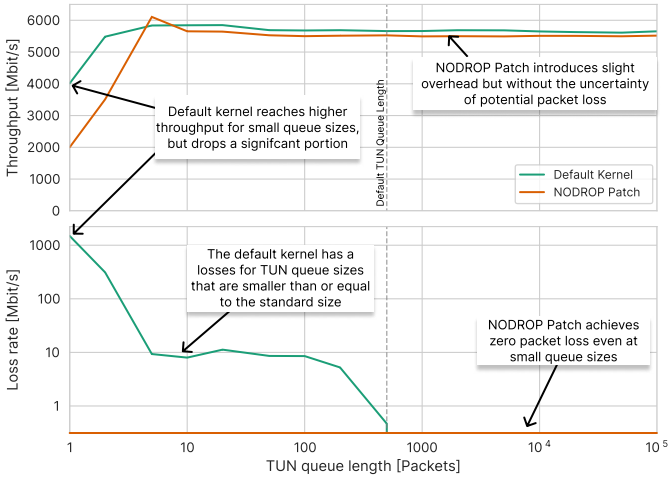

Fig. 4. Throughput and loss rate measurements for both the default TUN driver and the proposed NODROP Patch depending on the TUN queue size.

An increasing number of lost packets can be observed for TUN queue sizes equal to or smaller than the default TUN queue size of 500 packets, which is reflected in the loss data rate. Furthermore, the measurements of the NODROP Patch confirm that the suspected code was responsible for the observed packet drop behavior. No packet loss was recorded for any TUN queue size, even down to 1 packet. The results also show that the NODROP Patch achieves a consistent throughput level for TUN queue sizes greater than 10 packets. It has a negligible impact of around 4% compared to the default kernel. For TUN queue sizes smaller than 10 packets, a drop in throughput can be observed for both kernel versions, but while the NODROP Patch has no losses, the default kernel drops up to 1100 Mbit/s.

IV. Case Study: The NODROP Patch Improves User Space VPN Performance

Following the previously achieved successes, the next step was to apply the NODROP Patch to the problem of real-time video streaming via VPN from the motivation. In this case, a representative selection of VPN protocols is made. First, it was ensured that all these use the TUN driver for their user space implementation. The VPNs under consideration include the older but still widely used OpenVPN and the user space implementation of the new state-of-the-art WireGuard, the so-called WireGuard Go. In addition to the kernel implementation of Wireguard, which is mainly referred to, its user space implementation is of specific relevance nowadays. For compatibility reasons, most of the new mesh VPN solutions like Tailscale & Netbird [14] rely on this user space implementation. In addition, the multi-link solution SEAMLESS, which also relies on the TUN driver, is considered as part of the original motivation. Finally, the Linux kernel implementation of the WireGuard VPN is also validated as a benchmark. In general, the contestant protocols rely on similar principles. While WireGuard (Kernel/Go) and SEAMLESS rely on the User Datagram Protocol (UDP) transport protocol, OpenVPN can use both UDP and TCP. We only consider the UDP transport mode for the latter measurements for comparison. Furthermore, no specific configurations, aside from necessary key generation and endpoint configuration, were performed for the VPNs to observe near-default behavior, which refers to the standard settings of the VPNs without any additional optimizations.

Fig. 5 shows the evaluation setups for the following measurements. On the one hand, a laboratory setup (referred to as *Lab Setup*) consisting of a Lattepanda 3 Delta as a sender and a significantly faster Lattepanda Sigma as a receiver. As the receiver side is significantly faster than the sender, packet loss on the receiver side is minimized. The sender and receiver are connected over a Lattepanda Sigma that acts as an ethernet bridge and can emulate delay similar to the vSTING Approach [15]. On the other hand, a setup with two Virtual Private Server (VPS) is utilized, where the servers are connected over the Internet (referred to as *Real-World Setup*). The first server, acting as the sender, is located in Germany, while the other server is located in the United States, resulting in a Round Trip Time (RTT) of 120 ms between the servers. For both setups, we implement an orchestration unit, which controls the sender and receiver units and, in the laboratory setup, the ethernet bridge as well. On the sender unit, different user space applications, such as iperf3 for throughput measurements and a real-time UDP Videostream for teleoperation analysis, can be selected. The orchestration unit then selects the current VPN software under study and applies specific kernel settings, such as the NODROP Patch or varying TUN queue sizes of the sender.

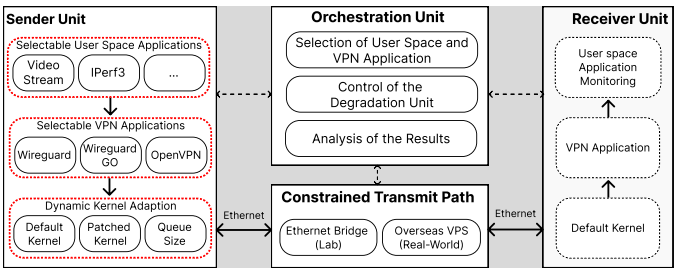

Fig. 5. Measurement architecture for the lab setup and real world measurements in regard to different user space applications, VPNs, and kernel parameters.

Any kernel-related parameters are set to the default on the sender side, while on the receiver side the socket buffer size was significantly increased which further minimizes packet loss on the receiver side. In the following chapters, we discuss the key results from our studies, beginning with the performance of a real-time video stream over VPNs with a reduced TUN queue to minimize worst-case latency.

A. Impact on Real-time Video Streaming

This measurement transmits a UDP-based video stream using VPNs over the lap setup. The requirements for the video stream are based on [1] with a data rate of 25 Mbit/s. The measurements were performed for all VPNs as well as a baseline measurement without a VPN. It was found that similar to the observed artifacts with the SEAMLESS approach, the quality of all transmissions over VPNs suffers from reduced TUN queue sizes, which are required for a low latency. Fig. 6 shows a screen capture of the impact for OpenVPN at a TUN queue size of 500 and 25 packets. A link to the resulting Videos after the transmissions with different VPNs and TUN queue sizes can be found in the caption of Fig. 6. While all VPNs perform decently for a TUN queue as low as 100 packets, all gain significant artifacts when reduced further. At 50 packets, WireGuard Go gains more minor artifacts, while OpenVPN and SEAMLESS suffer significant picture distortion, making

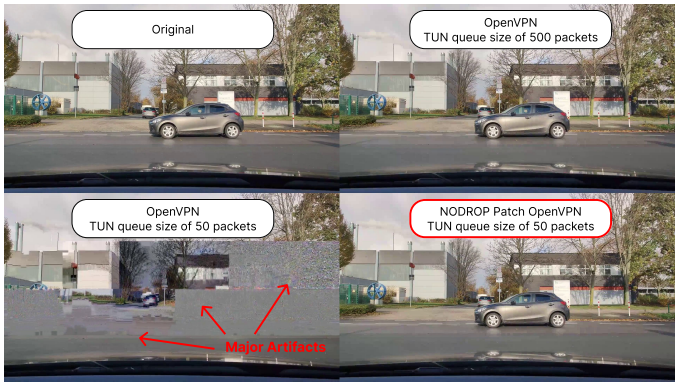

Fig. 6. Video transmissions for OpenVPN with default and reduced a TUN queue size (with and without NODROP). A complete comparison video containing the other VPNs can be found at https://youtu.be/oWlB_Xls2Ys.

them unsuitable for teleoperation applications. When reduced further, all VPNs are unusable for reliable transmission. In this case, the video data rate is significantly lower than the available throughput, but buffer overflow still occurs. It was concluded that the video encoder produces burst traffic on a video frame basis, overwhelming the reduced buffer and resulting in packet drops. This hypothesis is further empowered by examining the video stream performance with the NODROP Patch enabled for all VPNs. In this case, the TUN queue can be reduced to 1 packet without artifacts.

B. Impact on TCP-Throughput Performance

Next, the effects on the throughput of a TCP connection were considered. Like the UDP video stream, TCP connections are stream-based and react sensitively to packet loss. They require retransmissions for lost packets, halt the congestion window, and possibly reduce the throughput. First, we look at measurements using the Lab Setup. For low RTT, no latency was imposed, and only the TUN queue size was varied. Fig. 7 shows the achievable TCP throughput for the different VPN solutions with and without the NODROP Patch for different TUN queue sizes and a measurement without a VPN as a reference. It is initially noticeable that the throughput of all VPNs without the NODROP Patch strongly depends on the TUN queue size. Similar to the previous observations for the video stream, the data rate achieved drops sharply from a TUN queue size of less than 100 packets down to almost no throughput for a queue size of 1. Furthermore, as with the video stream measurements, the NODROP Patch drastically improves the performance of all VPNs for smaller TUN queue sizes up to similar data rates as the default kernel with queue sizes above 100. The optimum solution in this case would be the Wireguard Kernel module, which comes close to the practical maximum of a few percent.

Next, this behavior should also be validated again in a realistic setup. For this purpose, the setup with the overseas VPS, which has an RTT of around 120 ms, was chosen. The corresponding results, again depending on the TUN queue size, are shown in Fig. 8. In this case, the results differ from the measurements without additional latency. First of all, the generally achievable throughput is significantly lower.

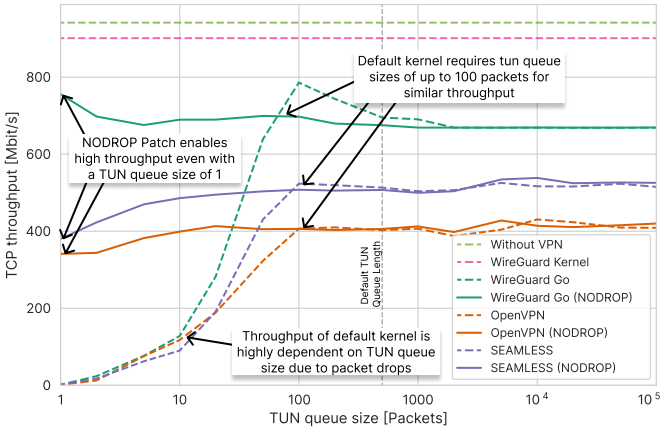

Fig. 7. Lab Setup: TCP throughput for different user space VPN solutions with and without the NODROP Patch. In addition both WireGuard kernel and the overall achievable throughput is shown for reference.

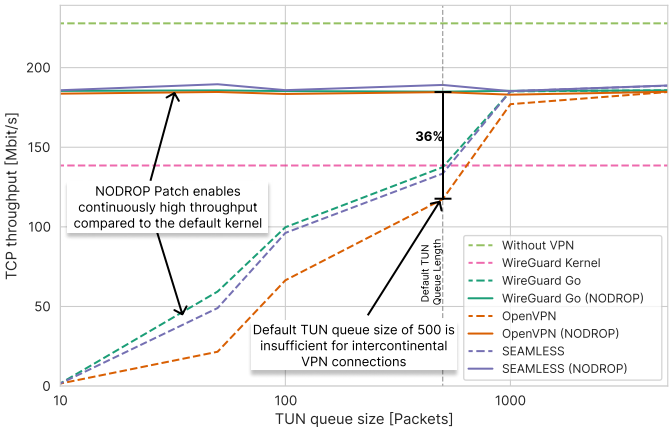

Fig. 8. Real-World Setup: TCP Throughput of the VPN solutions with and without NODROP Patch at a RTT of 120 ms.

Furthermore, the VPNs without the NODROP Patch require significantly larger TUN queue sizes (approx. 1000 packets) to achieve data rates similar to those of the patch. At the same time, the NODROP Patch enables consistently high data rates regardless of the TUN queue size. The patch improves the performance of all VPNs with the default tun queue size by up to 36% (OpenVPN). Interestingly, the Wireguard kernel does not perform ideally at the higher RTTs and falls behind the NODROP user space variants. After these measurements with increased latency showed substantial deviations from those without imposed latency, the next step was considering different latencies in the lab setup for the default tun queue size of 500 packets. The corresponding results for TCP throughput are shown in Fig. 9. Here, the strong dependence between the TCP data rate and the RTT can be seen for all methods. Due to the higher RTT, it takes longer for the transmitted packets to be acknowledged, which means that further data is initially held back. Initially, substantial variance in the data rates can be seen for all VPNs without the NODROP Patch, which increases for OpenVPN and SEAMLESS with increasing RTT. At the same time, the NODROP Patch improves the data rates drastically for all methods of a factor of up to 4.7x with a slight increase in variance for higher RTTs. Wireguard Kernel shows a consistent data rate depending on the RTT.

Due to the significantly reduced outliers with Wireguard Kernel, and with the NODROP Patch, it is reasonable to assume that extensive retransmissions cause performance losses for the unpatched VPNs due to TUN queue packet drops. To finally clarify this, the retransmissions for all procedures were analyzed for both vSTING and the overseas VPS scenario. Identical behavior was observed for all procedures using the TUN driver, which is represented in Fig. 10 by a time graph for the transmission of WireGuard Go in the overseas VPS setup. For the unpatched driver, it can be seen that the utilized TCP CUBIC algorithm [16] continuously attempts to increase the congestion window. However, the overloading of the TUN queue results in packet drops, which are reflected in an increase in retransmissions. TCP reduces the congestion window again accordingly. After a specific time, another attempt is made to increase the congestion window, resulting in retransmissions. In comparison, the NODROP Patch does not experience packet drops, which means that TCP maintains a continuously high

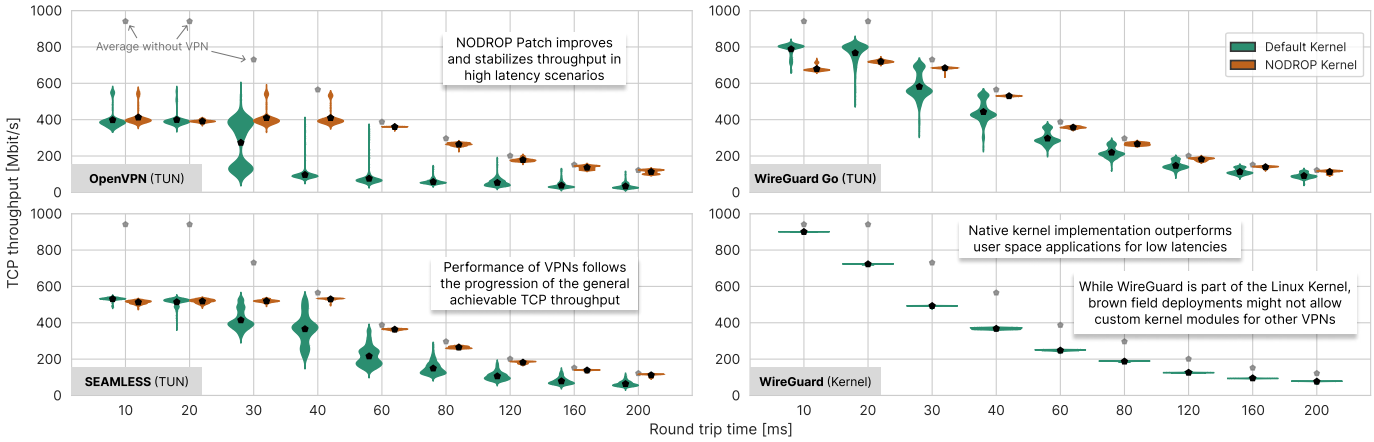

Fig. 9. Lab Setup: TCP throughput distribution of the considered VPN solutions for RTTs up to 200 ms. Note: Varying x-axis step width.

congestion window and achieves a correspondingly high and consistent throughput. No retransmissions occurred in the measurements for the NODROP kernel during the measurement period in multiple tests.

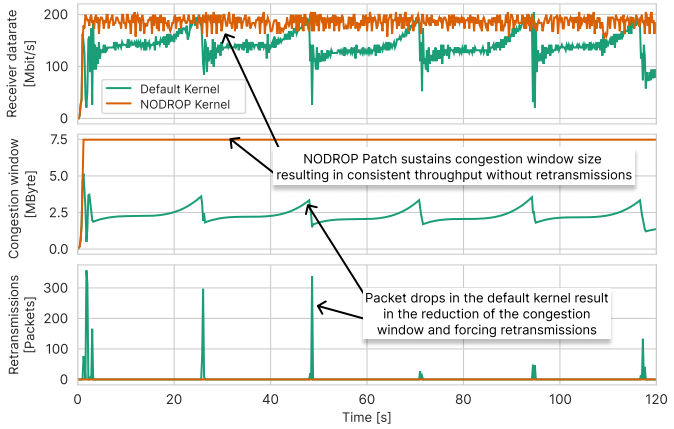

Fig. 10. Real-World Setup: Receiver datarate, congestion window size and retransmissions over time for WireGuard Go with both default TUN driver and NODROP Patch at a RTT of 120 ms.

V. Conclusion and Outlook

In this paper, a specific behavior of the Linux kernel TUN driver was identified, which leads to silent and preventable packet drops (up to 29 % for a reduced TUN queue size of 25 packets), significantly degrading the performance of real-time video streaming over VPNs. We proposed the open-source NODROP Patch to address this issue, allowing the TUN queue to be reduced from 500 down to even one packet, thereby drastically reducing the worst-case latency without impacting the video stream. At the same time, the Patch enables significant throughput improvements of up to 36% for TCP transmission for an intercontinental connection at a RTT of 120 ms. Aside from efforts to merge the NODROP Patch into the mainline kernel to make it broadly available to major distributions like Ubuntu and Fedora, future work will focus on the impact of the NODROP Patch on open 5G core networks, which also rely on the TUN driver. In addition, another focus will be on further improving the Linux sender stack.

Acknowledgment

This work has been funded by the German Federal Ministry of Research, Technology and Space (BMFTR) via the 6GEM research hub under funding reference 16KISK038 and is further supported by the project DRZ (Establishment of the German Rescue Robotics Center) under funding reference 13N16476.

References

[1] H. Schippers, T. Gebauer, K. Heimann, and C. Wietfeld, “RISE: Multi-Link Proactive Low-Latency Video Streaming for Teleoperation in Fading Channels,” in *Proc. IEEE VTC-Spring*, Oslo, Norway, 2025.

[2] T. Gebauer, M. Patchou, and C. Wietfeld, “SEAMLESS: Radio Metric Aware Multi-Link Transmission for Resilient Rescue Robotics,” in *Proc. IEEE SSRR*, 2023.

[3] S. Schippers and T. Gebauer, *NODROP Patch for Linux TUN Driver*, https://github.com/tudo-cni/nodrop, 2025.

[4] 5GAA, “Tele-Operated Driving (ToD): System Requirements Analysis and Architecture,” 5GAA Automotive Association, Munich, Germany, Tech. Rep., Sep. 15, 2021, p. 78.

[5] S. Neumeier, E. A. Walelgne, V. Bajpai, J. Ott, and C. Facchi, “Measuring the Feasibility of Teleoperated Driving in Mobile Networks,” in *Proc. Network Traffic Measurement and Analysis Conf. (TMA)*, 2019.

[6] M. Moniruzzaman, A. Rassau, D. Chai, and S. M. S. Islam, “High Latency Unmanned Ground Vehicle Teleoperation Enhancement by Presentation of Estimated Future through Video Transformation,” *J. of Intell. & Robotic Syst.*, vol. 106, no. 2, p. 48, Oct. 13, 2022.

[7] N. B. Truong, Y.-J. Suh, and C. Yu, “Latency Analysis in GNU Radio/USRP-Based Software Radio Platforms,” in *Proc. IEEE MILCOM*, 2013.

[8] M. S. Saud, H. Chowdhury, and M. Katz, “Heterogeneous Software-Defined Networks: Implementation of a Hybrid Radio-Optical Wireless Network,” in *Proc. IEEE WCNC*, 2017.

[9] F. Kaltenberger, A. P. Silva, A. Gosain, L. Wang, and T.-T. Nguyen, “OpenAirInterface: Democratizing innovation in the 5G Era,” *Computer Networks*, vol. 176, p. 107 284, 2020, ISSN: 1389-1286.

[10] The Linux Foundation, *Linux Kernel Driver for TUN/TAP*, https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/drivers/net/tun.c, Version 6.14, Accessed: May 22, 2025.

[11] W. KeHe, Z. Peng, C. WenChao, Z. XianKang, and C. AJun, “Tunneling SSL VPN Based on PF_RING,” in *Proc. IEEE ICSESS*, 2019.

[12] T. Shreedhar, N. Mohan, S. K. Kaul, and J. Kangasharju, “QAware: A Cross-Layer Approach to MPTCP Scheduling,” in *Proc. IFIP Networking Conf. and Workshops*, 2018.

[13] Y. Yin *et al.*, “Efficient Throughput and Loop-Free Routing: An Adaptive Second-Order Backpressure Algorithm,” in *Proc. IEEE INFOCOM WKSHPS*, 2024.

[14] V. Kjorveziroski, C. Bernad, K. Gilly, and S. Filiposka, “Full-mesh vpn performance evaluation for a secure edge-cloud continuum,” *Software: Practice and Experience*, vol. 54, no. 8, pp. 1543–1564, 2024.

[15] M. Patchou, J. Tiemann, C. Arendt, S. Boecker, and C. Wietfeld, “Realtime Wireless Network Emulation for Evaluation of Teleoperated Mobile Robots,” in *Proc. IEEE SSRR*, 2022.

[16] S. Ha, I. Rhee, and L. Xu, “CUBIC: a new TCP-friendly high-speed TCP variant,” *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, 2008.