Poster: Towards Open Wireless Time-sensitive Networking in Linux

Doğanalp Ergenç
ergenc@ccs-labs.org
EECS, TU Berlin
Berlin, Germany

Ahmed Abdulfattah
a.abdulfattah@tu-berlin.de
EECS, TU Berlin
Berlin, Germany

Ahmed Hasan Ansari
ansari@ccs-labs.org
EECS, TU Berlin
Berlin, Germany

Falko Dressler
dressler@ccs-labs.org
EECS, TU Berlin
Berlin, Germany

Abstract

The integration of IEEE 802.1 Time-sensitive Networking (TSN) with wireless technologies promises to support real-time applications over hybrid networks. Although TSN over Ethernet is well established, extending its capabilities to WiFi remains a technical challenge due to hardware and protocol differences. In this work, we integrate a core TSN scheduling mechanism, IEEE 802.1Qbv Time-aware Shaper (TAS), with WiFi interfaces in the Linux kernel. Our approach enables scheduling for mixed-criticality traffic over wired and wireless links using standard Linux tools. We provide our implementation and evaluation scenarios as open source to support further research in TSN-WiFi integration.

CCS Concepts

• **Networks → Link-layer protocols**.

Keywords

WiFi, TSN, Linux, Time-aware Shaper, TAPRIO

ACM Reference Format:

Doğanalp Ergenç, Ahmed Abdulfattah, Ahmed Hasan Ansari, and Falko Dressler. 2025. Poster: Towards Open Wireless Time-sensitive Networking in Linux. In *The 31st Annual International Conference on Mobile Computing and Networking (ACM MOBICOM '25), November 4–8, 2025, Hong Kong, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3680207.3765673

1 Introduction

Mission-critical systems have become highly heterogeneous interconnecting several stationary and mobile components. This necessitates the design of hybrid networks combining wired and wireless technologies to fulfill end-to-end quality of service (QoS) and reliability requirements for mixed-criticality traffic. On the wired side, IEEE 802.1 Time-sensitive Networking (TSN) is now the defacto standard to enable time-sensitive communication over Ethernet networks. On the wireless side, the current generation of WiFi introduces new features for reliable and real-time wireless communication. Although there is a growing research effort to enable cooperation between TSN and WiFi [1, 2], their coexistence introduces significant integration challenges due to hardware and protocol differences. In parallel, recent open-source projects provide new interfaces between TSN protocols and Linux systems to simplify implementation and configuration [3, 4]. Nevertheless, integrating TSN protocols over WiFi requires substantial engineering effort and adaptation of TSN configuration routines to the wireless domain. Currently, the available tooling for this is highly limited.

Our main contribution is the integration of the prominent TSN scheduling protocol, IEEE 802.1Qbv Time-aware Shaper (TAS), with WiFi interfaces in Linux. TAS enables scheduling for mixed-criticality traffic, and is already supported via the Linux Time-aware Priority Scheduler (TAPRIO) utility.[1] Since TAPRIO is currently only configurable on Ethernet interfaces, we extend the Linux kernel to enable shaping of egress traffic over WiFi interfaces. This allows: (i) the design of hybrid TSN-WiFi networks using standard Linux tooling and (ii) unified traffic scheduling managed by a single TSN controller across wired and wireless segments. We also release our implementation and several evaluation scenarios in a Linux-based network emulator as open source.[2]

[1]TSN for Linux, https://tsn.readthedocs.io/qdiscs.html
[2]The source code will be available at https://github.com/tkn-tub.

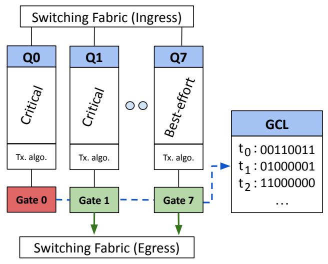


Figure 1: Overview of IEEE 802.1Qbv TAS.

2 Preliminaries

IEEE 802.1Qbv Time-aware Shaper: TAS schedules mixed-criticality Ethernet frames per their priority into time slots ($t_0, t_1..t_i$) of a schedule cycle. It models multiple priority queues with transmission gates controlled by a gate control list (GCL). For each time slot, the GCL can open multiple transmission gates according to its configuration. For example, in Figure 1, GCL is configured at the window t_1 to allow only TAS gate 0 and 7 to be open (green) and the other is closed (red). Per queue, a transmission selection algorithm decides on the frames to be forwarded first.

Linux Time-aware Priority Scheduler: TAPRIO is one of the queuing disciplines (qdiscs) in Linux to schedule mixed-criticality Ethernet traffic based on time and priority, directly corresponding with TAS. It processes socket buffers (`skb`), which is the fundamental data structure for network packets within the kernel. Each `skb` carries a priority field, which can be set by classifiers, and is used by the kernel's qdiscs for traffic prioritization and scheduling.

The TAPRIO command below sets up four traffic classes (`num_tc`) on the `if0` interface and maps `skb` priorities 0–3 to traffic classes 0–3 (`map`). Each class is assigned one transmission queue (`queues`). The schedule (`sched-entry`) defines two slots of 5 ms: the first allows transmission from classes 0 and 1 (`0x03`), and the second from classes 0 and 2 (`0x05`).

```
tc qdisc replace dev if0 parent root handle 10 taprio
  num_tc 4 map 0 1 2 3 0 0 0 0 0
  queues 1@0 1@1 1@2 1@3 base-time 1753168134
  sched-entry S 0x03 5000000
  sched-entry S 0x05 5000000
  clockid CLOCK_TAI
```

3 Design

Originally, TAPRIO only supports Ethernet interfaces with multiple hardware queues. Enabling it on WiFi necessitates modifications on several modules in the Linux kernel (see Figure 2). In this section, we highlight the necessary changes and describe the transmission process as following:

(1) The application packets (e.g., TCP or UDP) are encapsulated in `skb` and passed to TAPRIO.

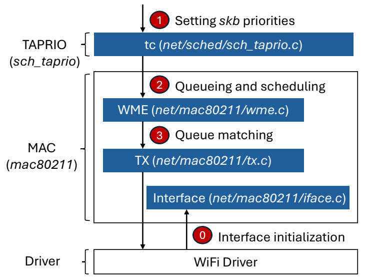


Figure 2: TAPRIO-enabled WiFi stack in Linux kernel.

(2) TAPRIO pushes them into the priority queues according to their priority matching, and forwards them to MAC in their scheduled slots.
(3) On the kernel-side, packets are further matched with WiFi queues and passed to the transmission module.
(4) They are passed on to the driver for transmission.

MAC Modifications: `mac80211` provides implements WiFi MAC in Linux kernel. `iface.c` implements the creation of 802.11 interfaces. Normally, the recent kernels do not expose the hardware queues of WiFi equipment to the higher-level traffic control utilities but leaves the autonomy to the WiFi equipment. Instead, we modified it to allocate multiple transmission queues at the MAC level, accessible by TAPRIO for scheduling. The number of allocated queues are limited with the available hardware queues in the WiFi interface.

However, this modification still necessitates one-to-one mapping between scheduler queues and those in the WiFi interface. We further modified `wme.c` and `tx.c` to bypass the existing WiFi QoS management and consider queue mapping enforced by TAPRIO (i.e., `skb->queue_mapping`) instead. Here, `WME` is responsible for QoS management, specifically matching between packet priorities and WiFi access categories (ACs). `TX` performs transmission-related operations before the frames are handed over the WiFi driver. In both modules, we utilize the variable (`skb->tc_skip_classify`) set in TAPRIO module and described in the next section.

Scheduler Modifications: `sch_taprio` implements TAPRIO. We modified it to mark scheduled packets by setting `skb->tc_skip_classify` to 1. Normally, this flag is used by intermediate functional block (IFB) devices to avoid further QoS management when the packets are already processed. We repurpose this to override WiFi QoS management at `WME`.

Another change is in budgeting. TAPRIO calculates a budget to determine how many frames can be sent during a schedule time slot, based on the gate opening duration and bitrate of the network interface. Although TAPRIO can detect the speed of Ethernet interfaces via kernel functions, the specifications of WiFi interfaces are not accessible. Therefore, we set the budget based on the effective bitrate of 35 Mbit/s for 802.11g as used in our experiments.

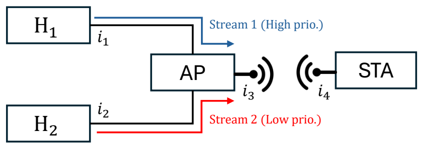


Figure 3: Experiment topology.

Note that effective use of the scheduler requires disabling the default contention-based channel access in WiFi. This can be achieved by setting backoff counters to minimum while retaining channel sensing to avoid collisions.

4 Evaluation

We evaluate our implementation in the Linux-based network emulator Mininet-WiFi.[3] It creates virtual Ethernet and WiFi interfaces and emulates wireless medium in Linux. It uses `mac80211_hwsim` kernel module to emulate WiFi drivers and does not perform channel contention. Therefore, transmissions are performed based on TAPRIO scheduling.

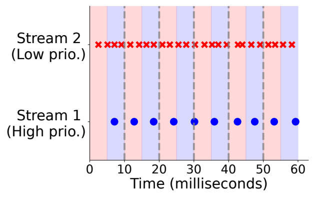
For the experiments, we consider the topology in Figure 3. For wireless interfaces (i_3 and i_4), we set 802.11g parameters with 54 Mbit/s bitrate. For traffic model, two wired Ethernet hosts (H_1 and H_2) send streams at the rate of 2 Mbit/s and 5 Mbit/s to the station (STA), respectively. They are distinguished with different VLAN PCP values. To shape the transmission of these streams, we configure egress i_3 interface of the access point (AP) with a 10 ms schedule cycle divided into two 5 ms time slots for high and low priority.

Figure 4 shows the frame transmission timeline on i_3 (of AP) with and without TAPRIO configuration. The timelines are further divided into 5 ms time slots for high (Stream 1, blue) and low (Stream 2, red) priority. As Figure 4a shows, when AP is not scheduled with TAPRIO, the frames are transmitted without a certain pattern and sent at the transmission rate of i_3. In contrast, in Figure 4b, the TAPRIO-enabled AP transmits high- and low-criticality frames at their designated time slots for every schedule cycle.

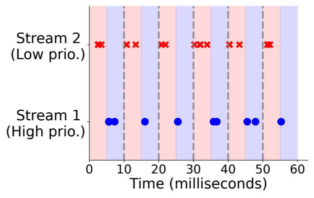
Figure 5a shows the rolling average (lines) and 95^{th} percentile (areas) of end-to-end latency between H_1/H_2 and STA over the consecutive packets. While most packets are delivered under 10 ms aligned with the schedule cycle, there are still several latency peaks due to two main reasons. Firstly, talkers cannot send packets with perfectly steady intervals and this causes some frames to miss their transmission slots at AP. Secondly, the stochastic modeling of the wireless link between AP and STA induces additional delay.

To eliminate the impact of the talker-induced irregularities, we configure i_1 and i_2 interfaces of the talkers with TAPRIO. This enables H_1 and H_2 to send (a burst of) packets every 5 ms, aligning with the configuration at AP. As seen in Figure 5b, all transmissions are achieved under 10 ms latency, still with a certain jitter due to wireless link characteristics.

[3]Mininet-WiFi, https://mininet-wifi.github.io/

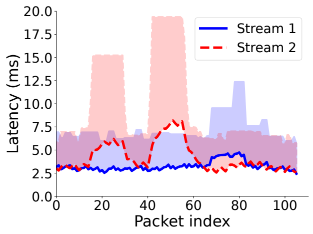


(a) Without TAPRIO

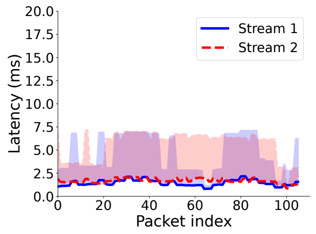


(b) With TAPRIO

Figure 4: Transmission timeline on AP.



(a) TAPRIO only at AP



(b) TAPRIO at AP and talkers

Figure 5: End-to-end latency.

5 Conclusion

In this work, we integrate the Linux queuing discipline TAPRIO with WiFi to enable time-sensitive transmission of mixed-criticality traffic over wireless links. This facilitates unified scheduling in hybrid TSN-WiFi networks using open-source Linux tools. Our evaluation demonstrates improved latency predictability in an emulation environment. Future work includes real-world deployment on Linux-based WiFi devices and developing a wireless-aware scheduler.

Acknowledgments

This work was supported by the OWTSN project as a part of TARGET-X program under grant number 101096614.

References

[1] Toni Adame, Marc Carrascosa-Zamacois, and Boris Bellalta. 2021. TSN in IEEE 802.11be: On the Way to Low-Latency WiFi 7. *MDPI Sensors* 21, 15 (July 2021), 4954. https://doi.org/10.3390/s21154954

[2] Dave Cavalcanti, Carlos Cordeiro, Malcolm Smith, and Alon Regev. 2022. WiFi TSN: Enabling Deterministic Wireless Connectivity over 802.11. *IEEE Communications Standards Magazine* 6, 4 (Dec. 2022), 22–29. https://doi.org/10.1109/MCOMSTD.0002.2200039

[3] Wei Quan, Wenwen Fu, Jinli Yan, and Zhigang Sun. 2020. OpenTSN: An Open-source Project for Time-sensitive Networking System Development. *CCF Transactions on Networking* 3 (2020), 51–65. https://doi.org/10.1007/s42045-020-00029-8

[4] Filip Rezabek, Marcin Bosk, Georg Carle, and Jörg Ott. 2023. TSN Experiments Using COTS Hardware and Open-Source Solutions: Lessons Learned. In *IEEE PerCom 2023, Workshops*. IEEE, Atlanta, GA, 466–471. https://doi.org/10.1109/PerComWorkshops56833.2023.10150312