Time-Limited Software Firewall Based on DPDK Supporting TSN and DetNet Traffic

Lukas Bechtel*†, Markus Schramm*, Michael Menth§, Tobias Heer*†
*University of Applied Sciences Esslingen, Germany, {lukas.bechtel,markus.schramm,tobias.heer}@hs-esslingen.de
§Chair of Communication Networks University of Tuebingen, Germany, menth@uni-tuebingen.de
†Belden Inc., Neckartenzlingen, Germany

***Abstract*—The convergence of IT and OT networks introduces strict latency and security requirements, especially in virtualized industrial environments. While TSN and DetNet provide bounded-latency traffic delivery, traditional software firewalls break determinism due to variable rule evaluation times. Hence, we propose a time-limited firewall design that limits per-packet rule evaluation time to a fixed budget, ensuring deterministic processing even under high load. To preserve security despite partial rule checks, we propose that a deferred filtering stage verifies and, if necessary, retroactively corrects earlier forwarding decisions. We implement this design in a software firewall prototype and evaluate it under maximum packet rate. The system guarantees limited latency for all packets, maintains high throughput, and ensures eventual security consistency, all without requiring specialized hardware.**

***Index Terms*—Industry 4.0, TSN, Time-Sensitive Networking, Security, Network Security, Network Segmentation, IoT**

I. Introduction

Modern industrial automation systems depend on time-critical communication. Control operations, real-time sensor data, and safety functions rely on the timely and predictable delivery of messages. Even small deviations in timing can cause instability, performance degradation, or safety risks. This challenge intensifies with the adoption of edge computing and local data centers since time-critical traffic is no longer restricted to isolated network segments. Instead, traffic passes through shared network segments, routers, and firewalls. Best-effort traffic with high data rates also uses these components, which makes isolating traffic and preserving timing guarantees increasingly difficult. Time-Sensitive Networking (TSN) and Deterministic Networking (DetNet) address the need for time-critical communication in shared networks and across multiple network segments. These standards ensure deterministic timing behavior if network designers model the timing of the network with the knowledge of the latencies of all individual components, i.e., switches, routers, and firewalls.

Today, software firewalls do not perform time-deterministic filtering and thus, break the deterministic guarantees required by TSN and DetNet environments [1]. To address this issue and forward traffic at wire speed without packet loss, we propose a time-limited software firewall architecture that enforces a per-packet processing budget. The firewall assigns each packet a time budget. If the firewall cannot decide within this budget, i.e., does not reach a matching rule, it immediately applies a default action, such as ACCEPT or DROP. This design prioritizes latency bounds over complete ruleset evaluation. Figure 1 compares the state-of-the-art behavior with packet loss and latencies above the time limit with the target latency limited by the time limit and no packet loss.

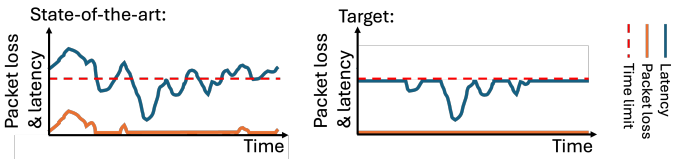


Fig. 1. Limiting filter duration to support static latencies for critical traffic.

To maintain security, a deferred filtering mechanism asynchronously re-evaluates forwarded packets and corrects any misclassification for future traffic. This guarantees bounded latency per packet while preserving security by comparing the executed decision to the correct rule in the ruleset. As the performance of a firewall varies with different packet rates, we introduce the concept of a dynamic time budget. With that, the initially configured maximum budget is dynamically adjusted so that the firewall can process all packets in a timely manner.

In summary, our contributions are as follows:

- We analyze the impact of high packet rates on the available time for filtering in software firewalls.
- We present a time-limited firewall design that bounds the per-packet filter duration to a time budget.
- We introduce and discuss a mechanism to limit the introduced security tradeoff.
- We enhance the model with a dynamic time budgeting mechanism that adapts to the packet rate.

Our prototype, implemented using FD.io VPP based on the Data Plane Development Kit (DPDK), demonstrates the feasibility of this approach. It delivers high-throughput, low-latency firewalling with adaptive filtering depth and post-hoc verification. This architecture enables software firewalls to operate within deterministic networking environments, combining real-time responsiveness with security enforcement.

II. Related Work

Chomsiri et al. improved rule evaluation by organizing rules into trees [2], enhancing session tables [3], and combining

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 528745080 - FIP 68 and "FHP: Qualifizierung und Entwicklung des professoralen Personals der Hochschule Esslingen für zukunftsweisende Themen" (FKZ: 03FHP115) as part of the "FH-Personal", funded by the BMBF and MWK Baden-Württemberg. The authors alone are responsible for the content of the paper.

Accepted for publication in Proceedings of *30th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Porto, Portugal, September 9-12, 2025

stateless filtering with automatic rule reordering [4]. These approaches focus on throughput but do not address latency bounds. Rovniagin et al. integrated the GEM algorithm into iptables for faster matching [5], but its complexity remains unsuitable for real-time guarantees. Trabelsi et al. proposed a single-hash session table [6] and later optimized rule and field ordering to reduce processing for repeated traffic and early drops [7]. El-Atawy et al. reduced operations using statistics-based filters [8] and prefiltering based on boolean logic [9]. Wang et al. reordered rules dynamically to shorten matching paths [10]. Unlike these works, our approach enforces strict per-packet latency by limiting rule evaluation time and recovering security through deferred filtering.

III. Design

In this section, we present the architecture of the time-limited firewall, detailing its core mechanisms for enforcing latency bounds and preserving security.

A. Overview and Background

The duration of the filtering for a single packet depends on the index of the rule a packet matches. Rules in the front of the ruleset cause a smaller delay than rules in the back of the ruleset. We introduce the mechanism for upper-bounding the filter duration in Section III-B. To limit the attack surface introduced by the early forwarding, we present the deferred filtering in Section III-C. Software firewalls process packets sequentially. Hence, consecutive packets will only be filtered if the filtering of the previous packet is finished. As a result, the duration of packet processing and filtering must not exceed the time between the arrival of two packets. For example, at 1 Gbit/s with 64-byte Ethernet frames, packets arrive approximately every 0.67 µs. As the required bound for the filter duration depends on the packet rate, we present the dynamic time limit in Section III-D.

Class-based queues generally improve performance for critical traffic as well. However, the firewall stack VPP used in this paper supports batch processing of up to 256 packets, which is only applicable if a single queue is used.

B. Time-Limited Filtering

The core of the presented approach is a firewall design that enforces deterministic latency by bounding the time required for rule filtering. This is achieved by applying a fixed time budget to the filtering of each packet and aborting rule evaluation when the budget is exhausted.

The architecture consists of two stages. In the first stage, a fast classification ruleset assigns each packet a time budget and a default action (cf. Figure 2 A). The goal is to have very coarse rules, e.g., based on VLAN Priority Code Point (PCP), EtherType, or IP protocol fields. This mapping allows administrators to express latency expectations for different traffic classes without fine-grained differentiation between packets. Hence, these classification rules are simple and fast to evaluate, as there are only a few rules.

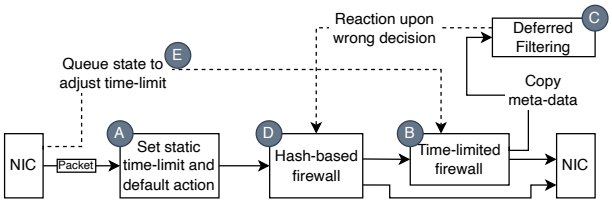


Fig. 2. Design overview for the time-limited firewall.

In the second stage, the packet enters the main rule evaluation pipeline, which performs conventional packet filtering based on fields such as IP addresses, ports, and protocols (cf. Figure 2 B). These rules are more granular and specific to the traffic. Hence, rulesets can contain more than 1,000 rules for firewalls in the backbone. To enforce time constraints, the firewall checks if the time budget is exceeded every n^{th} rule, where n is configurable. Checking after every n^{th} rule only instead of after every rule reduces the slowdown of filtering, as the costly time check is required less often. If the elapsed processing time exceeds the assigned budget, rule evaluation is aborted, and the configured default action is applied, i.e., the packet is forwarded or dropped. This ensures that no packet processing exceeds the allocated latency budget, regardless of ruleset complexity or runtime load.

This mechanism guarantees that each packet is processed within a strict upper latency bound, regardless of traffic conditions or the size of the ruleset. It ensures compatibility with TSN and DetNet latency guarantees.

C. Deferred Filtering

Time-limited filtering introduces the possibility of incomplete rule evaluation. Specifically, an attacker can send packets matching a drop rule at the end of the ruleset. Due to the static limit, the filtering is aborted, and the packet is forwarded. To address this attack vector, the firewall includes a deferred filtering mechanism that continues rule matching asynchronously after stopping rule evaluation and forwarding the packet due to the time budget being exceeded (cf. Figure 2 C). This deferred path enables a secondary layer of inspection that catches, logs, and reports mismatches without delaying real-time traffic.

When a packet exceeds its time budget, the firewall applies the default action, i.e., forwards or drops the packet. In both cases, the firewall creates a metadata record that includes:

- Header fields from the original packet (excluding payload to reduce the required memory),
- The index of the next rule that would have been evaluated,
- The default action taken.

This metadata is passed to a deferred filtering unit, implemented as a separate task, which runs on a dedicated CPU core or a remote processor. This deferred filtering unit resumes evaluation of the rule to the end of the ruleset.

Suppose the deferred filtering detects a mismatch between the taken and configured action, for example, a malicious packet was incorrectly accepted. In that case, the firewall must react immediately and prohibit further packets of this malicious traffic, e.g., by adding a rule to the beginning of the

ruleset. However, the firewall is required to continue filtering with the configured time limits and filtering at least the most important rules at the beginning of the ruleset. Hence, we introduce a hash-based firewall (cf. Figure 2 D) in front of the regular sequential filtering. If the deferred filter detects a mismatch, the hash of the 5-tuple of the IP header is inserted into the hash table, combined with the desired action, i.e., ACCEPT or DROP. For every incoming packet, the firewall executes a lookup first and only enters the sequential filtering if no entry in the hash table matches. This hash table introduces a static delay for every packet, independent of the table size.

D. Dynamic Time Budgeting

In industrial networks, packet rates can vary, ranging from high peak rates during backup phases to low rates during regular control operations. Typically, a software firewall is capable of filtering and forwarding packets quickly enough to meet the timing requirements of industrial applications. However, if the packets arrive faster than the firewall can filter them, the queues fill, latency increases drastically, and the firewall drops packets. The static limits ensure that a specific rate of filtering is achievable. However, larger rates also cause increased latencies and packet loss for the time budget mechanism. An intuitive solution is to configure the time limit to align with the maximum packet rate to be expected, e.g., for 1 Gbit/s with 64-byte packets, to have a time budget of 600 ns. However, during phases with a lower packet rate, it would be sufficient for the firewall to use a larger time budget and thus, filter more rules before forwarding the packet. Therefore, we introduce the dynamic time budget, which uses the length of the ingress queue to adjust the time budget (cf. Figure 2 E). With this mechanism, the configuration of the initial time limit (cf. Figure 2 A) can use larger time limits and larger network loads reduce this limit dynamically to avoid increased latencies and packet loss.

IV. Security Discussion

Enforcing deterministic latency in software firewalls introduces a tradeoff between timely forwarding and comprehensive rule evaluation. This tradeoff creates new attack vectors, particularly during periods of high load when packets may be forwarded based on default actions before full rule evaluation completes. In this section, we discuss the protocol-aware configuration to mitigate this security risk.

The impact of a misclassified packet depends on its transport protocol and application context. UDP and industrial Layer 2 protocols such as PROFINET or OPC UA PubSub can carry payloads for the control process in every packet. Hence, falsely accepting a packet has a direct impact on the control application. In contrast, TCP-based connections require a three-way handshake, providing a natural barrier against immediate misuse. Specifically, the deferred filtering has one complete round-trip time to react to misclassified packets.

To mitigate protocol-dependent risks, the firewall supports protocol-aware configuration of time budgets (cf. Figure 2 A). Administrators can assign larger budgets to protocols that lack handshake mechanisms and can deliver data immediately. Protocols with a handshake can be forwarded more progressively, using the deferred filtering during the handshake and the hash-table (cf. Figure 2 D) as a security measure.

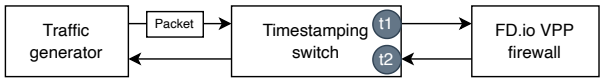


Fig. 3. Testbed Architecture.

V. Evaluation

In this section, we present the evaluation of the applicability of the design to limit the filter duration.

A. Testbed for the Evaluation

We evaluate the time-limited software firewall on an off-the-shelf industrial PC using a wired 1 Gbit/s Ethernet testbed. The firewall runs on Ubuntu 24.04 LTS and is based on FD.io VPP with the current version 25.02, which is built on top of the DPDK. In baseline tests, VPP outperformed other firewall stacks. We configure a constant CPU frequency and disable all power-saving features to minimize jitter.

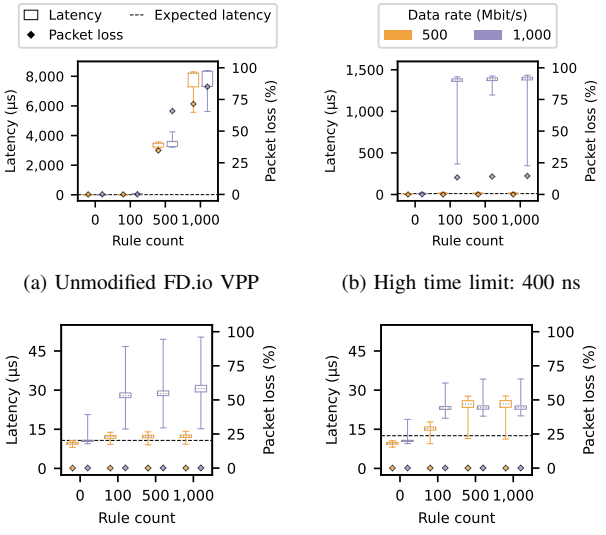
Figure 3 shows the testbed architecture. The traffic generator sends UDP packets with a configurable data rate and packet size. A timestamping switch between the traffic generator and the firewall inserts egress (*t1*) and ingress (*t2*) timestamps into each packet. This allows latency measurements with a precision of less than 20 ns by calculating the difference between *t1* and *t2*. The measured latency includes the complete packet transmission, processing, and filtering. For this evaluation, we use 64-byte packets at a constant 1 Gbit/s rate. This creates a worst-case load for the filtering. We send two parallel streams with different latency budgets per measurement:

- One critical stream with 10 Mbit/s.
- One best-effort stream that fills the remaining bandwidth.

We vary the number of firewall rules from 0 to 1,000, using both static time budgets and dynamic budgeting that adapts based on system load. Each configuration runs 100 times. We measure end-to-end latency and packet loss.

B. Time Budget Granularity

Checking the remaining time budget on every rule evaluation requires additional processing. Consequently, the firewall evaluates fewer rules within the same time frame. In the following, we report the rule evaluation speed of the firewall in rules per µs where higher values indicate better performance. The firewall used in this evaluation processes 243 rules per µs in the unmodified state. We evaluated the impact of checking the time budget when the firewall checks it every n^{th} rule evaluation. The overhead of evaluating the time budget depends on the load on the system. For example, 64-byte packets at 100 Mbit/s result in 44 rules per µs when checking after every rule and 103 rules per µs after every 16^{th} rule. With this granularity, the firewall continues to filter for 15 rules, even though the limit is completely consumed, resulting in a weaker level of precision. At the same time, the forwarding of



(a) Unmodified FD.io VPP

(b) High time limit: 400 ns

(c) Low time limit: 200 ns

(d) Adaptive time limit: < 2 µs

Fig. 4. Latency with different configurations (64-byte, 1 Gbit/s).

the firewall without rules has a jitter of 12.2 µs. As the filtering of one rule costs 4.1 ns, the additional 15 rules cost 61.5 ns, i.e., less than the forwarding jitter. Therefore, externally, this timing granularity is not measurable.

C. Throughput and Packet Loss

In this section, we evaluate the latency and packet loss of the firewall. Figure 4 shows the results in four configurations: unmodified VPP, high time limits, low time limits, and adaptive time limit. Each subfigure combines latency (scale on the left) and packet loss (scale on the right). For the latency analysis, we use boxplots with the whiskers indicating the 99% percentile. We vary the number of firewall rules from 0 to 1,000 and evaluate at two data rates: 500 Mbit/s and 1 Gbit/s.

The unmodified VPP setup (cf. Figure 4a) cannot handle large rule sets, neither at 1 Gbit/s nor at 500 Mbit/s. Latency remains low for a small number of rules, but exceeds 4,000 ms at 500 rules. Packet loss approaches 100 percent.

In the high time limit configuration (cf. Figure 4b), filtering is aborted if the filtering duration exceeds the configured time limit of 400 ns. Hence, the firewall reduces the latency and packet loss for 500 and 1,000 rules. However, for 100 rules at 1 Gbit/s, the overhead of the modifications increases the latency and packet loss compared to the unmodified version. Specifically, due to the overhead checking for the time limit, the evaluation of the rules takes longer than in the unmodified version, and the overall processing time is larger than the time between consecutive packets.

The low time limit configuration (cf. Figure 4c) ensures low and stable latency across all rule counts and both data rates. Packet loss remains zero. The firewall applies default actions early for both data rates, after a mean of 41 rules, enforcing a strict time budget, specifically for 500 Mbit/s. However, a larger CPU load at 1 Gbit/s generates a static offset to the expected latency and a jitter of up to 35.1 µs.

The adaptive configuration (cf. Figure 4d) begins with an initial time budget of 2 µs and adjusts dynamically, i.e., reduces it with growing queue sizes. This results in low latency and minimal packet loss, even at high rule counts. The load on the CPU for many rules causes a jitter of 15 µs. Additionally, the median latency consistently exceeds the expected latency by up to 12.2 µs. For 500 Mbit/s, the firewall evaluates a mean of 142 rules, using all available time compared to 39 rules with the low time limit. For 1 Gbit/s, the firewall reduces the evaluated rules to a mean of 33, also reducing the jitter.

In summary, the low and adaptive time limits effectively reduce the filter budget, enabling wire-speed forwarding without packet loss. However, the load on the CPU causes a latency jitter of 35.1 µs with a static time budget, compared to 16.5 µs with the adaptive budget and 12.2 µs without the firewall.

VI. Conclusion

Deterministic latency is a key requirement in time-sensitive networks. However, conventional software firewalls have a variable rule evaluation time. We proposed a novel filtering mechanism that introduces a tradeoff between security and determinism by enforcing a fixed per-packet evaluation time budget. To ensure correctness, we introduced deferred filtering, which verifies and corrects earlier decisions while simultaneously improving rule coverage through learning. An adaptive timing mechanism dynamically adjusts the time budget to system load, thereby increasing the filter duration while preserving the time limit. The prototype demonstrates that latency bounds can be upheld in software, even under high traffic rates of 1 Gbit/s, with minimal compromise to security.

References

[1] L. Wüsteney, M. Menth, R. Hummen, and T. Heer, "Impact of Packet Filtering on Time-Sensitive Networking Traffic," in *IEEE International Conference on Factory Communication Systems (WFCS)*, Linz, Austria, Jun. 2021.

[2] X. He, T. Chomsiri, P. Nanda, and Z. Tan, "Improving cloud network security using the Tree-Rule firewall," *Future Generation Computer Systems*, vol. 30, 2014.

[3] T. Chomsiri, X. He, P. Nanda, and Z. Tan, "A Stateful Mechanism for the Tree-Rule Firewall," in *International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, 2014.

[4] ——, "Hybrid Tree-Rule Firewall for High Speed Data Transmission," *IEEE Transactions on Cloud Computing (TCC)*, vol. 8, no. 4, 2020.

[5] D. Rovniagin and A. Wool, "The Geometric Efficient Matching Algorithm for Firewalls," *IEEE Transactions on Dependable and Secure Computing (TDSC)*, vol. 8, no. 1, 2011.

[6] Z. Trabelsi and S. Zeidan, "Enhanced Session Table Architecture for Stateful Firewalls," in *IEEE International Conference on Communications (ICC)*, 2018.

[7] Z. Trabelsi, L. Zhang, S. Zeidan, and K. Ghoudi, "Dynamic traffic awareness statistical model for firewall performance enhancement," *Computers & Security*, vol. 39, 2013.

[8] A. El-Atawy, T. Samak, E. Al-Shaer, and H. Li, "Using Online Traffic Statistical Matching for Optimizing Packet Filtering Performance," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2007.

[9] A. El-Atawy, E. Al-Shaer, T. Tran, and R. Boutaba, "Adaptive Early Packet Filtering for Defending Firewalls Against DoS Attacks," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2009.

[10] W. Wang, H. Chen, J. Chen, and B. Liu, "Firewall Rule Ordering Based on Statistical Model," in *International Conference on Computer Engineering and Technology (ICCET)*, 2009.