TUM SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

eBPF-Assisted Relays for Multimedia Streaming

Daniel Alexander Antonius Pfeifer

TUM SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

eBPF-Assisted Relays for Multimedia Streaming

eBPF-Unterstützung für Multimedia-Streaming-Netzknoten

Author:	Daniel Alexander Antonius Pfeifer
Supervisor:	Prof. Dr. Ing. Jörg Ott
Advisor:	Mathis Engelbart, M.Sc.
Submission Date:	15.08.2024

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, 15.08.2024 Daniel Alexander Antonius Pfeifer

Acknowledgments

Abstract

Contents

1 Introduction

1.1 Citation Examples

Citation [Lam94]. Citation [Int24b]. Citation [Int24a]. Citation [Rod24].

1.2 Research Question

TODO

1.3 Scope

TODO

1.4 Structure of this Thesis

TODO

1.5 Source Code Repositories

TODO

2 Background and Related Work

2.1 QUIC

The Transmission Control Protocol (TCP) has been used as the backbone of the internet for more than 40 years. It has been designed to be reliable and to provide a connection-oriented way of transmitting data, but the modern environment of the internet with its need for increasing throughput make it hard for TCP to keep up. Limitations in the design and resulting issues like head-of-line blocking have raised demand for a newly designed protocol that can keep up with the modern internet. The 'Quick UDP Internet Connections' protocol, short QUIC protocol, is a transport layer protocol built on top of UDP that is designed to be reliable, cryptographically secure and more performant than TCP. It was intended to be the successor of TCP and it has its origins at Google before the standardization by the IETF began in 2016. QUIC, partly because it operates both in user- and kernel-space, has been designed to allow for a more rapid deployment cycle than TCP. Similar to TCP it is a connection-based protocol that uses TLS for encryption. As of beginning of 2024, QUIC already made up 7.7% of all internet traffic. With the main driver being Google's own services, like YouTube, where QUIC is already used by default. Together with HTTP/3, which is already implemented in more than 90% of all web browsers, this number is likely to increase even further.

2.1.1 Connections and Streams

Since QUIC is a connection-based protocol, some initial overhead to establish a connection is needed. However the design incorporates some features that aim for a more efficient way of establishing connections, e.g. by using 0-RTT (zero round trip time) handshakes. Latency improvements like the 0-RTT handshake however come at the cost of security, since that opens the door for replay attacks. Another part where QUIC tries to optimize connection management is the use of streams. Streams are designed to be lightweight and can be opened without the need of a handshake. It even goes as far that a single packet can contain the opening of a new stream, stream data as well as the closing of the stream. This allows for new techniques to improve data transmission and will also be part of the fast-relay setup in this thesis. Aside from streams, apparent since QUIC is based on UDP, it is also possible to send data via unreliable datagrams.

This further improves versatility of the protocol and allows for new ways of optimizing data transmission.

2.1.2 quic-go and moqtransport

The implementation of the proposed fast-relay setup will be based on the quic-go library, which provides a pure Go implementation of the QUIC protocol as specified in the standards RFC-9000, RFC-9221 as well as some others which are not that important in this thesis. Together with a modified version of the quic-go library, the fast-relay implementation will also use the moqtransport library. This library brings the 'Media over QUIC' (MoQ) protocol to Go and will be used as a media transport protocol when looking at the impact of fast-relays on adaptive real-time video streaming. The MoQ protocol is being standardized by the IETF since July 2023 and has yet to be finalized.

2.1.3 QUIC and Fast Relays

The QUIC protocol will be a fundamental part of the fast-relay setup in this thesis, yet the ideas used to make relays faster is not limited to QUIC and can be extended to other protocols as well. QUIC is chosen as an example protocl due to its increasing popularity which offers big potential in early adoption and deployment of fast-relays. Besides that, the existing implementations of QUIC related standards provide a good starting point for an implementation, despite the difficulties that the heavy encryption of QUIC brings with it. To mitigate missing technologies, mainly for offloading QUIC decryption and encryption onto hardware, the existing protocol libraries can also be modified easily to simulate any needed behavior.

2.2 eBPF

In 1992 a technology called 'Berkeley Packet Filter' (BPF) was introduced into the Unix kernel. By using BPF it is possible to attach a small BPF-program to some pre-defined hook points in the network stack of the kernel and filter packets there in a stateless manner. This provided more efficiency since the packets did not need to be copied into userspace anymore but could directly be processed in the kernel. One downside to such an approach however is that BPF-programs are limited by the so-called 'BPF-verifier' which needs to check every BPF-program for safety e.g. to avoid infinite loops or access to invalid memory from withing kernel space. Today, the initial technology of BPF has evolved into 'extended BPF' (eBPF) and allows for more versatile use cases.

2.2.1 eBPF Hook Points

The Linux kernel offers several hook points where eBPF-programs can be attached to. Two main ones are one to attach eBPF-programs to the Traffic Control (TC) subsystem and another one to attach them to the eXpress Data Path (XDP) subsystem. The XDP hook, which is directly located in the NIC-driver, lies lower in the network stack than the TC-hook, which is located in the link-layer. Despite being higher up in the network stack, the TC-hook has the big advantage that it offers ingress and egress processing while the XDP-hook is available for ingress processing only. This makes the XDP-hook suboptimal for the implementation of fast-relays since they heavily rely on processing packets at egress, after those were redirected from ingress. Figure 3.1 illustrates again the relative positions of the TC and XDP hook points in the network stack.

2.2.2 Traffic Control Queuing Disciplines

The Linux Traffic Control Subsystem uses Queuing Disciplines (qdiscs) to define how packets are handled. TODO

2.2.3 eBPF Verifier

TODO

2.2.4 Important eBPF Concepts

TODO

2.2.5 eBPF and Fast Relays

TODO

2.3 Adaptive Media Streaming

TODO

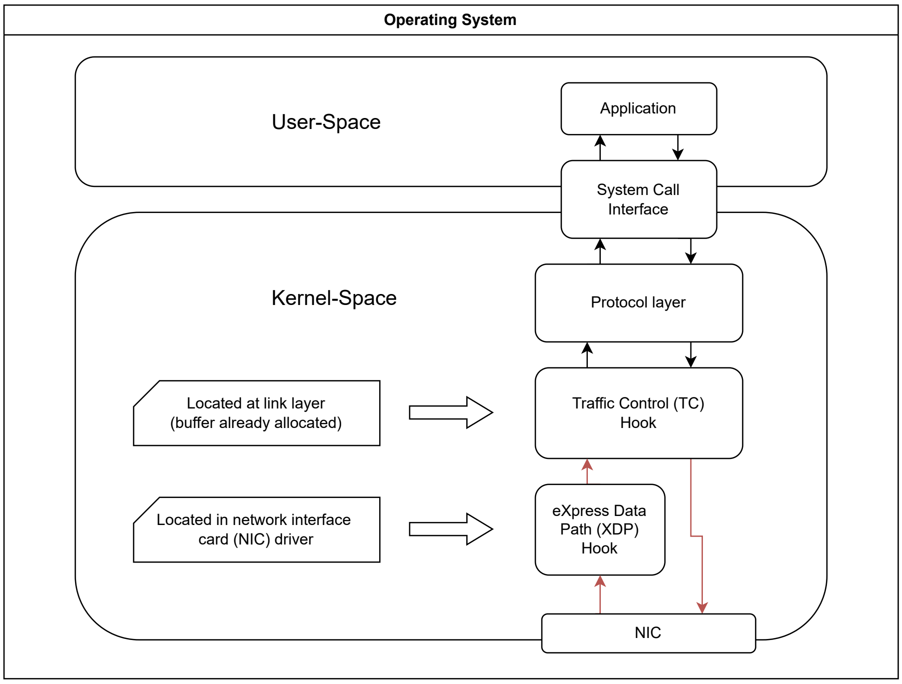

Figure 2.1: Abstracted view of Traffic Control (TC) and eXpress Data Path (XDP) hook points in the Linux kernel network stack. The red loop indicates the 'short-cut' that is utilized by the fast relay. TC hook allows redirection directly to egress while XDP hook is only available for ingress processing.

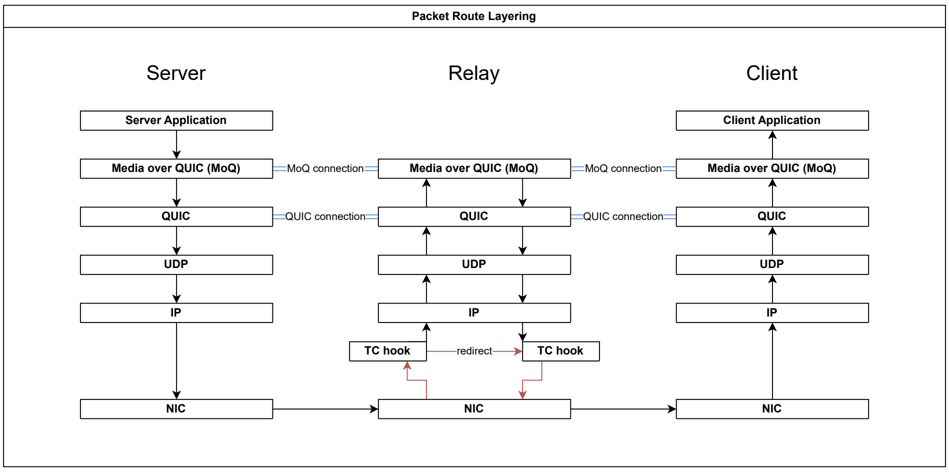

Figure 2.2: Conventional layers of a network stack for client, server and relay. The red loop indicates again the 'short-cut' that is utilized by the fast relay and based on eBPF packet-forwarding. This avoids the need for the packet to traverse the entire network stack of the relay up to the userspace.

3 Fast Relays

3.1 QUIC Adaptions

TODO

3.2 eBPF Setup

TODO

In order to make the congestion control algorithm that is running in userspace usable we need to inform the QUIC library about the forwarded packets. This again happens via BPF maps and a separate go routine that continuously polls new entries in the map and processes them. Entries are then added to the packet history to allow the receipt of ACKs. Besides that, the congestion control algorithm will be informed about the forwarded packet in order to be able to react to potential congestion events.

3.3 User Space Avoidance

TODO

3.4 Packet Filtering and Dropping

TODO

3.5 Client Congestion

TODO

3.6 Subscription and State Management

TODO

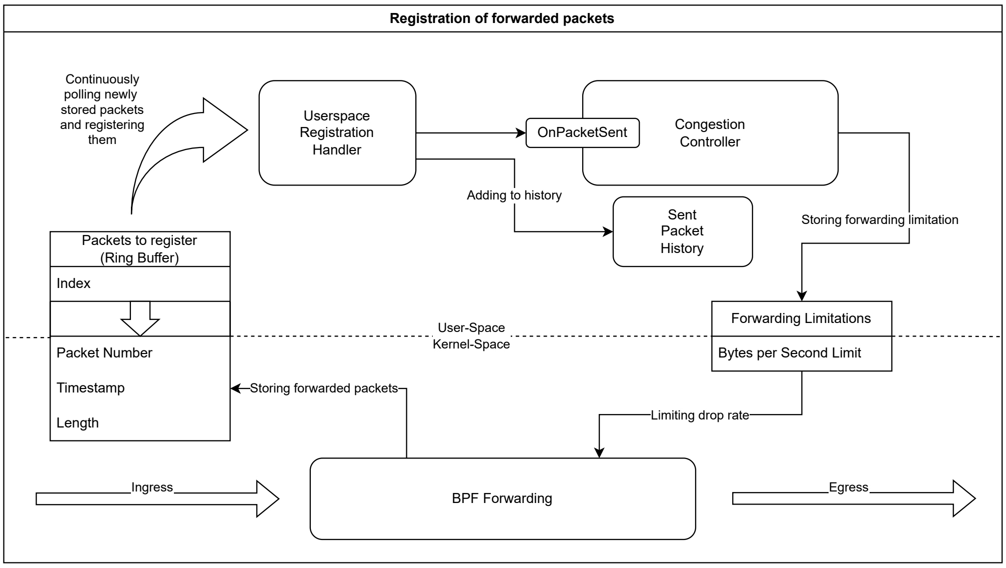

Figure 3.1: Internal setup for registering forwarded packets as well as incorporating forwarding limitations for the BPF program.

3.7 Relay Caching

TODO

3.8 Compatibility

TODO

4 Testing

5 Future Work

6 Conclusion

List of Figures

List of Tables

Bibliography

[Int24a] Internet-Engineering-Task-Force. *Media over QUIC Transport*. 2024.

[Int24b] Internet-Engineering-Task-Force. *QUIC: A UDP-Based Multiplexed and Secure Transport*. 2024.

[Lam94] L. Lamport. *LaTeX : A Documentation Preparation System User's Guide and Reference Manual*. Addison-Wesley Professional, 1994.

[Rod24] L. Rodriguez. *The Future of the Internet is here: QUIC Protocol and HTTP/3*. 2024.