eBPF-Assisted Relays for Multimedia Streaming

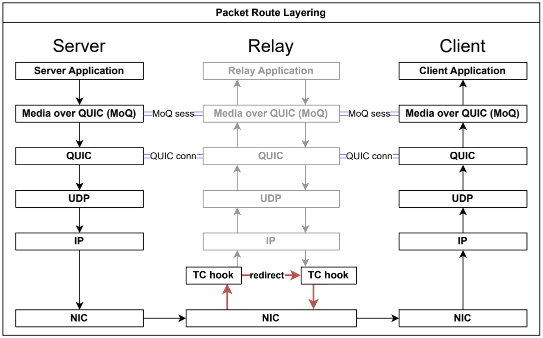
Daniel Alexander Antonius Pfeifer

Technical University of Munich

August 20, 2024

Motivation

- Shorten Critical Path
- Avoid Network Stack Traversal
- Reduce Forwarding Delay

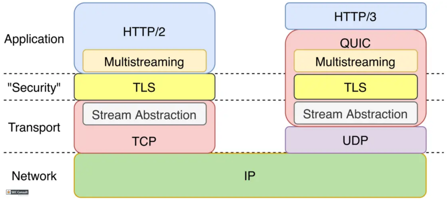


Research Question

- *Improve relay performance by using eBPF technology?*
    - *Remove userspace packet-processing from critical path?*
    - *Handle packet en- and decryption?*
    - *Communication between userspace and the eBPF program?*
    - *Generalize to support other protocols?*

QUIC

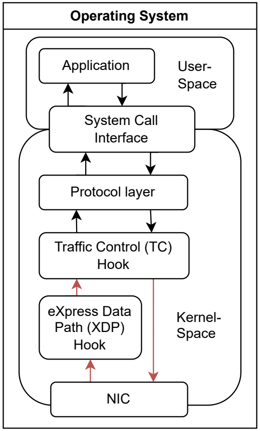
- Started by Google as *Quick UDP Internet Connections*
- Standardized by IETF
- Fast Development Cycle since Userspace Implementation
- Gets rid of Issues like Head-of-Line Blocking



Source: https://sec-consult.com/blog/detail/better-dont-be-too-quick/

eBPF

- Kernel-Internal Virtual Machine
- Used for Packet Filtering and Tracing
- Multiple Hook-Points in the Kernel (e.g. XDP and TC)
- Userspace Communication via Maps



QUIC Adaptations

- Turn off en- and decryption
- Priorities for packets
- Public endpoint for packet registration
- Function pointer additions for eBPF state handling
  - Relay developer defines functions for eBPF map access
  - Called within quic-go if defined

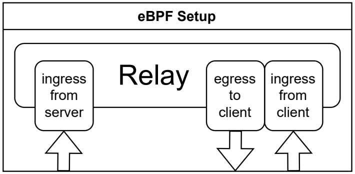
Public Endpoint for Packet Registration

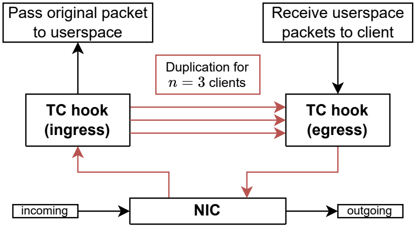
```go
go func(conn quic.Connection) {
    /* ... */
    for {
        /* ... */
        packet := common.RetrieveNextPacketFromMap()
        conn.RegisterBPFPacket(packet)
        /* ... */
    }
}(conn)
```

Listing 1: Packet registration within relay code.

eBPF Setup

- Three eBPF Programs
    - Client ingress (client registration)
    - Server ingress (packet duplication and forwarding)
    - Client egress (state management)

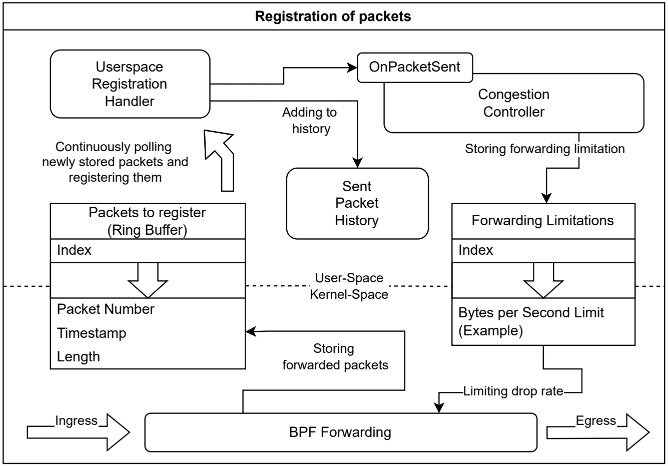



Pass original packet to userspace
Receive userspace packets to client
Duplication for $n = 3$ clients
TC hook (ingress)
TC hook (egress)
incoming
NIC
outgoing

Userspace Synchronization

- Number of clients
- Connection state (e.g. connection-id, id-translations, etc.)
- Incoming packet information (e.g. timestamp, etc.)
- Priority drop threshold for a connection
- Congestion control updates

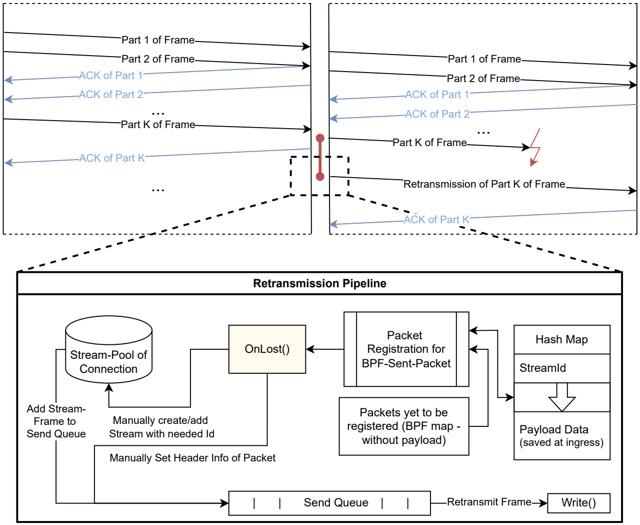
Userspace Synchronization cont.



Packet Retransmission

- Retransmission happen at stream level
- Relay might not have correct stream state
- Client needs **all** parts of a frame for correct media display

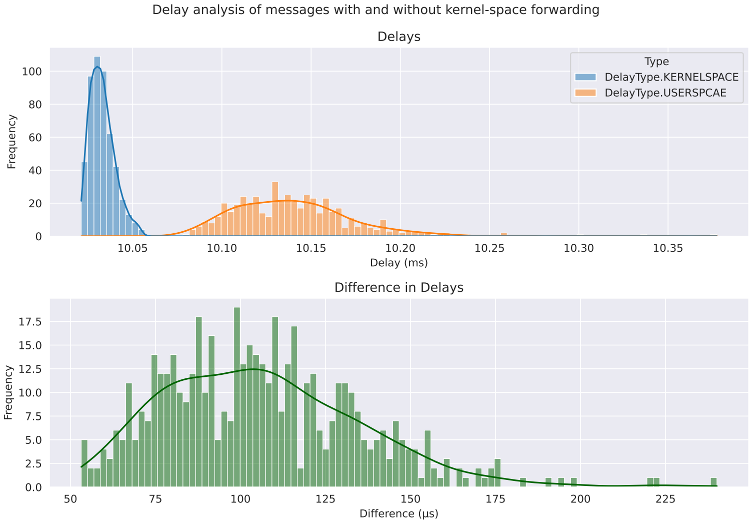
Packet Retransmission cont.


Part 1 of Frame
Part 2 of Frame
ACK of Part 1
ACK of Part 2
...
Part K of Frame
ACK of Part K
...
Part 1 of Frame
Part 2 of Frame
ACK of Part 1
ACK of Part 2
...
Part K of Frame
Retransmission of Part K of Frame
ACK of Part K
Retransmission Pipeline
Stream-Pool of Connection
OnLost()
Packet Registration for BPF-Sent-Packet
Hash Map
StreamId
Payload Data (saved at ingress)
Packets yet to be registered (BPF map - without payload)
Add Stream-Frame to Send Queue
Manually create/add Stream with needed Id
Manually Set Header Info of Packet
Send Queue
Retransmit Frame
Write()

Test Setup

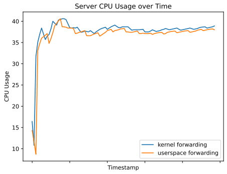
- Single machine setup (delay reduction only due to different kernel processing)
- Separate namespaces for client, relay, and server
- Artificial delay between client and relay for packet registration

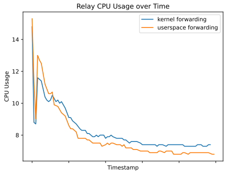
Test Results Delay Reduction

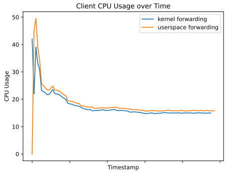


Test Results CPU Usage

- No Impact on CPU Usage
- Fewer System Calls
  - Mainly due to reduced Userspace Synchronization







System Calls

- Example Stream of 30 Seconds
- Overall System Calls
  - Userspace forwarding: 296132 calls
  - eBPF forwarding: 225674 calls
  - Reduction of 24%
- *futex*
  - Reduction of 34%
  - 21666 calls instead of 32940
- *nanosleep*
  - Reduction of 42%
  - 14293 calls instead of 24716
- *epoll_wait*
  - Reduction of 67%
  - 11289 calls instead of 34149

Conclusion

- Delay reduction via eBPF forwarding
- More application specific relay code needed
- No impact on CPU usage

Future Work

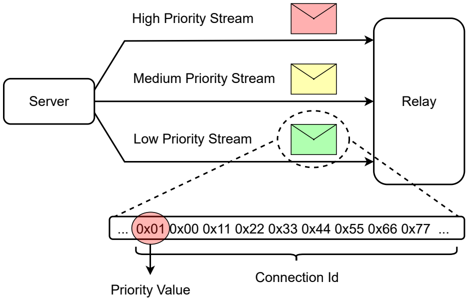
- Hardware offloading of en- and decryption
- Expand to other protocols
- Prototype completion
  - Congestion control
  - Physical setup for testing

That’s it!

Any Questions?

Packet Priorities

- One priority per stream
- Saved in connection-id
- Additional connection-id retirement constraint



Function Pointer Additions

```
/* Function pointer call within actual quic-go code */
if packet_setting.ConnectionIdUpdateBPFHandler != nil /* &&
    potentially other conditions */ {
    packet_setting.ConnectionIdUpdateBPFHandler(connId.Bytes(),
        uint8(connId.Len()), p.connection)
}
```

Listing 2: Function-pointer addition to the quic-go library.

```
/* Function pointer signature definition within additional
    config file */
ConnectionIdUpdateBPFHandler func(id []byte, l uint8, conn
    QuicConnection) = nil
```

Listing 3: The signature will be defined within the library itself.

Function Pointer Additions

```
/* Definition of the function within the local relay code */
func localUpdateConnectionId(id []byte, l uint8, conn
    packet_setting.QuicConnection) {
    /* handle the connection update by interacting with the eBPF
        program */
}

/* Providing the function to the quic-go library */
func main() {
    /* ... */
    packet_setting.ConnectionIdUpdateBPFHandler =
        localUpdateConnectionId
    /* ... */
}
```

Listing 4: An example of how the addition looks on the relay side.

Test Setup

```
 ----------------------         ---------------------------------------        ----------------------
|   Server namespace   |       |            Relay namespace            |      |   Client namespace   |
|     ____________     |       |    ______________   ______________    |      |     ____________     |
|    |192.168.10.1|    |       |   | 192.168.10.2 | | 192.168.11.2 |   |      |    |192.168.11.1|    |
|____|___veth0____|____|       |___|____veth1_____|_|____veth2_____|___|      |____|____veth3___|____|
            \                            /                 \                             /
             \                          /                   \                           /
              \                        /                     \                         /
               \                      /                       \                       /
                \ __________________ /_                      __\ ___________________ /
                /veth0-br|     |veth1-br\                   /veth2-br|      |veth3-br\
                |                    ___|                   |___                     |
                \_____v-net-0_______|NAT/                   \NAT|_________v-net-1____/
                        /              \                     /                \
               ip: 192.168.10.10        \                   /         ip: 192.168.11.10
               net: 192.168.10.0/24      \   ___________   /          net: 192.168.11.0/24
                                           /             \
                                          |   enp1s0f0    |
                                           \ ___________ /
                                                  |
                                                  |
                                              (INTERNET)
```