eBPF-Assisted Relays for Multimedia Streaming

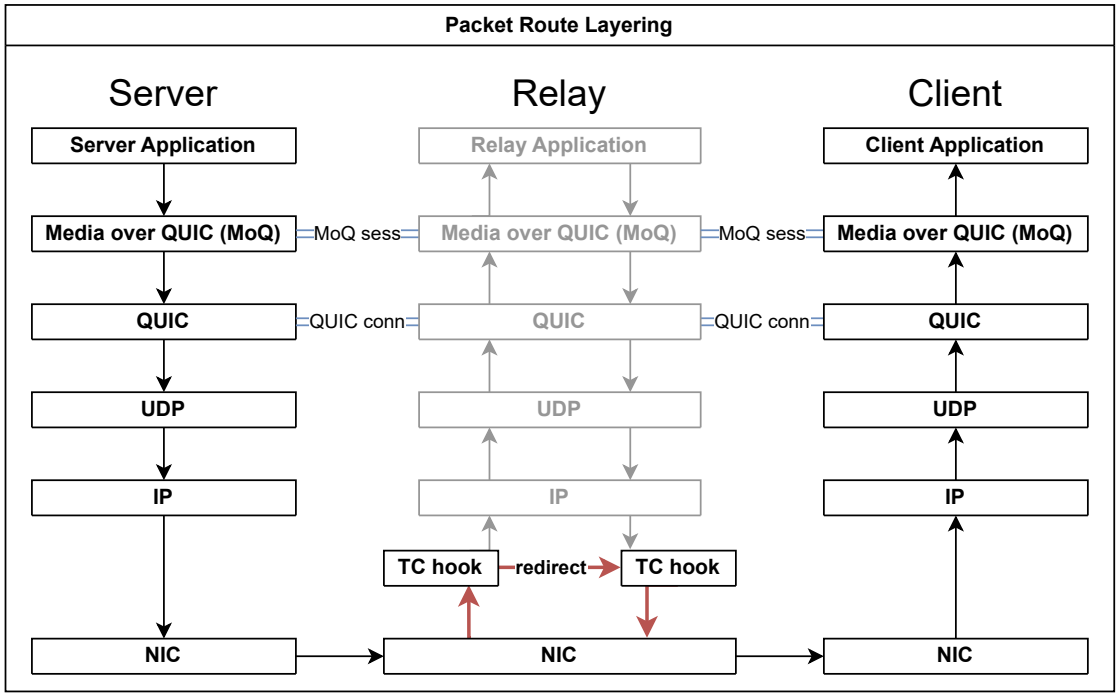
Daniel Alexander Antonius Pfeifer

Technical University of Munich

August 21, 2024

Motivation

- Shorten critical path
- Avoid network stack traversal
- Reduce forwarding delay

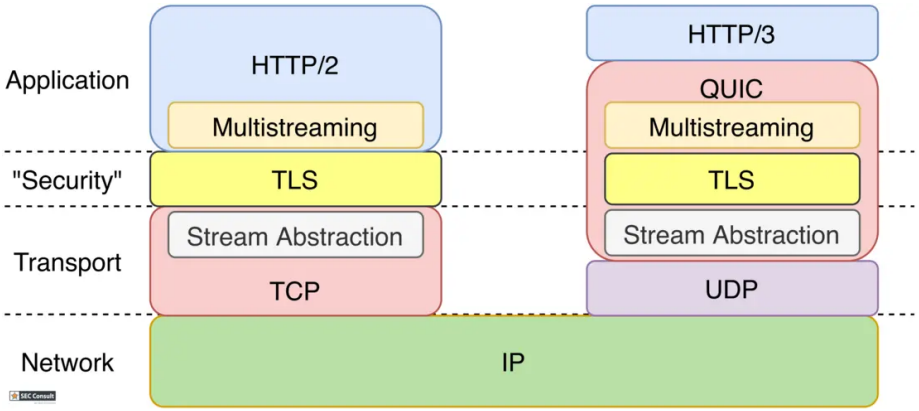


Research Question

- *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

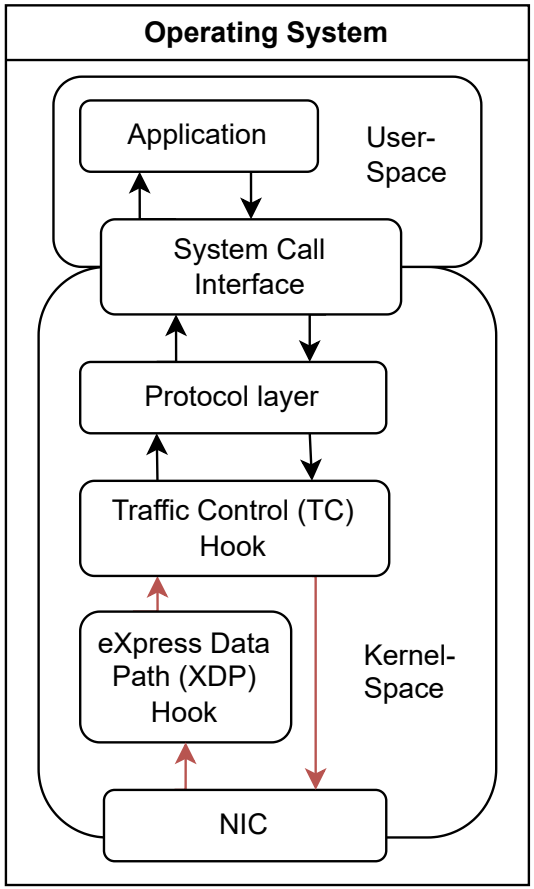
- Started by Google as *Quick UDP Internet Connections*
- Standardized by IETF
- Fast development cycle due to userspace implementation
- Gets rid of issues like head-of-line blocking



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

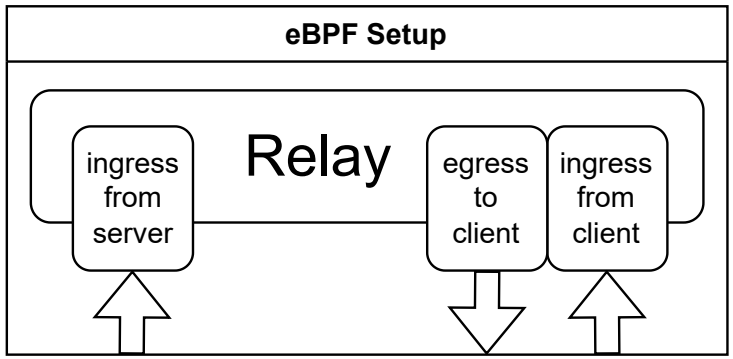
Public Endpoint for Packet Registration

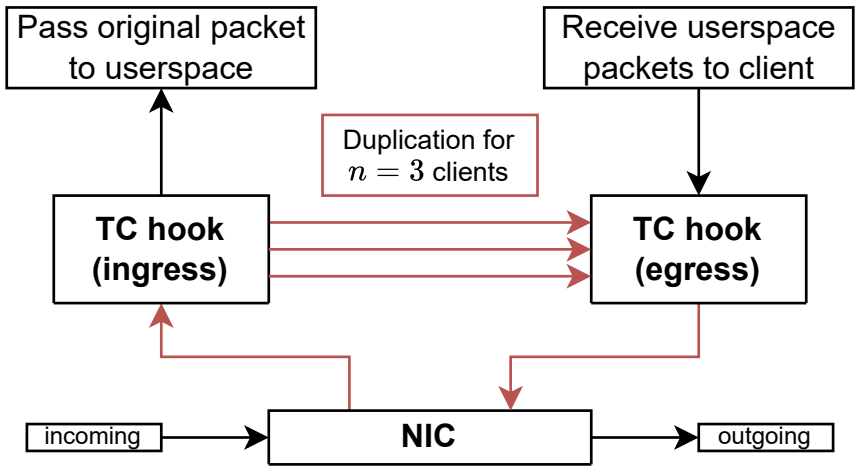
```
1 go func(conn quic.Connection) {
2     /* ... */
3     for {
4         /* ... */
5         packet := common.RetrieveNextPacketFromMap()
6         conn.RegisterBPFPacket(packet)
7         /* ... */
8     }
9 }(conn)
```

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)

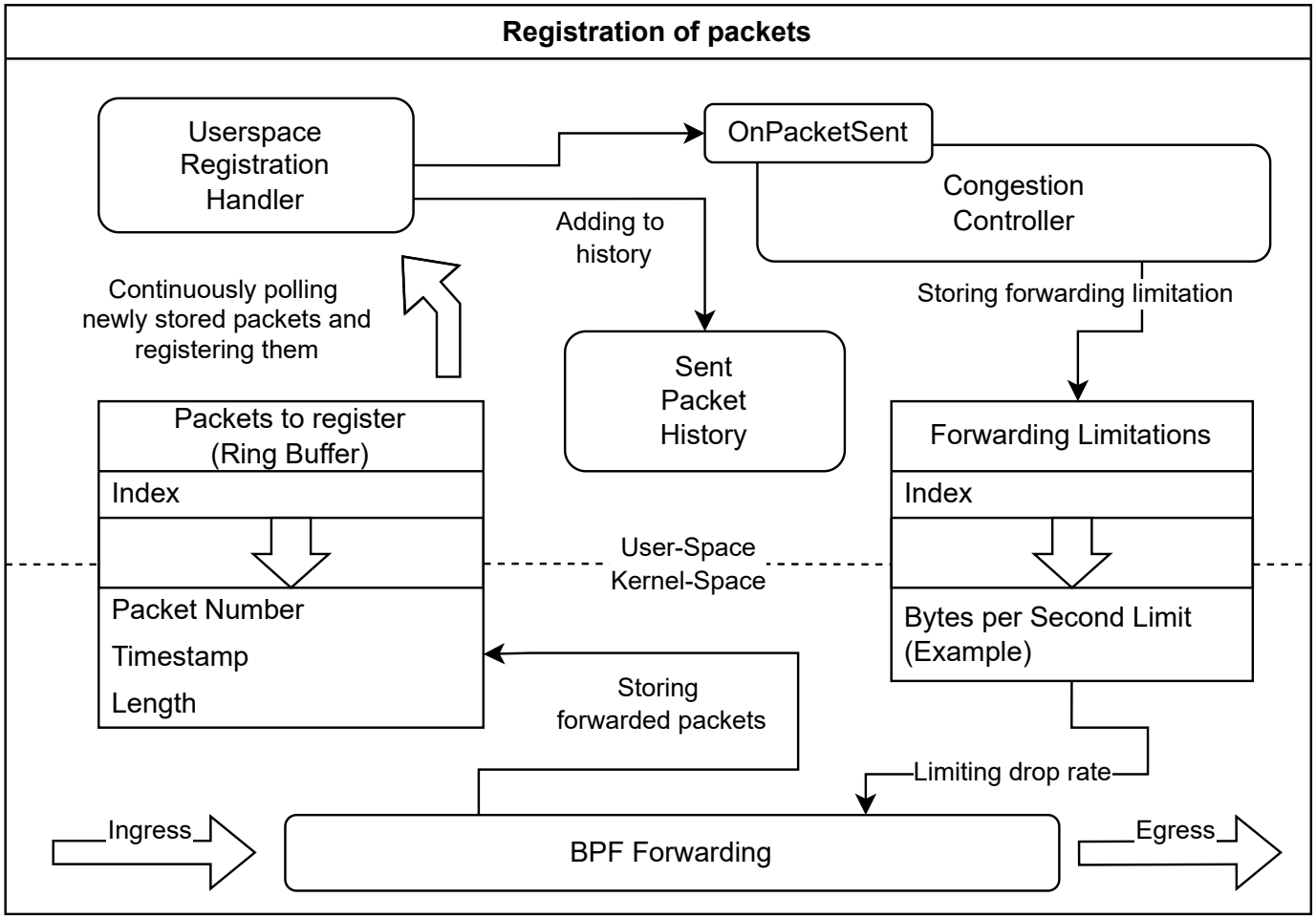




Userspace Synchronization

- 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

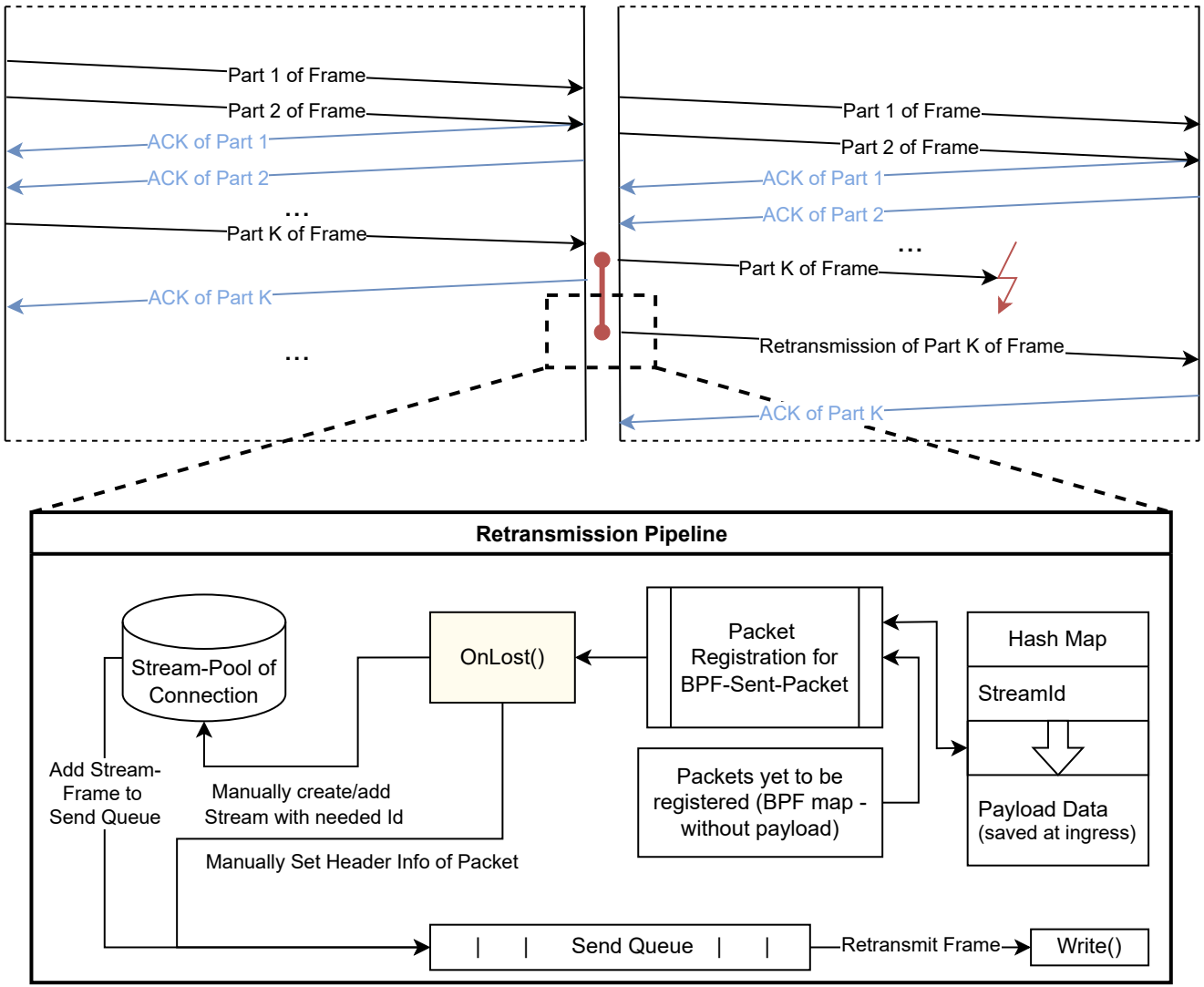
Userspace Synchronization cont.



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

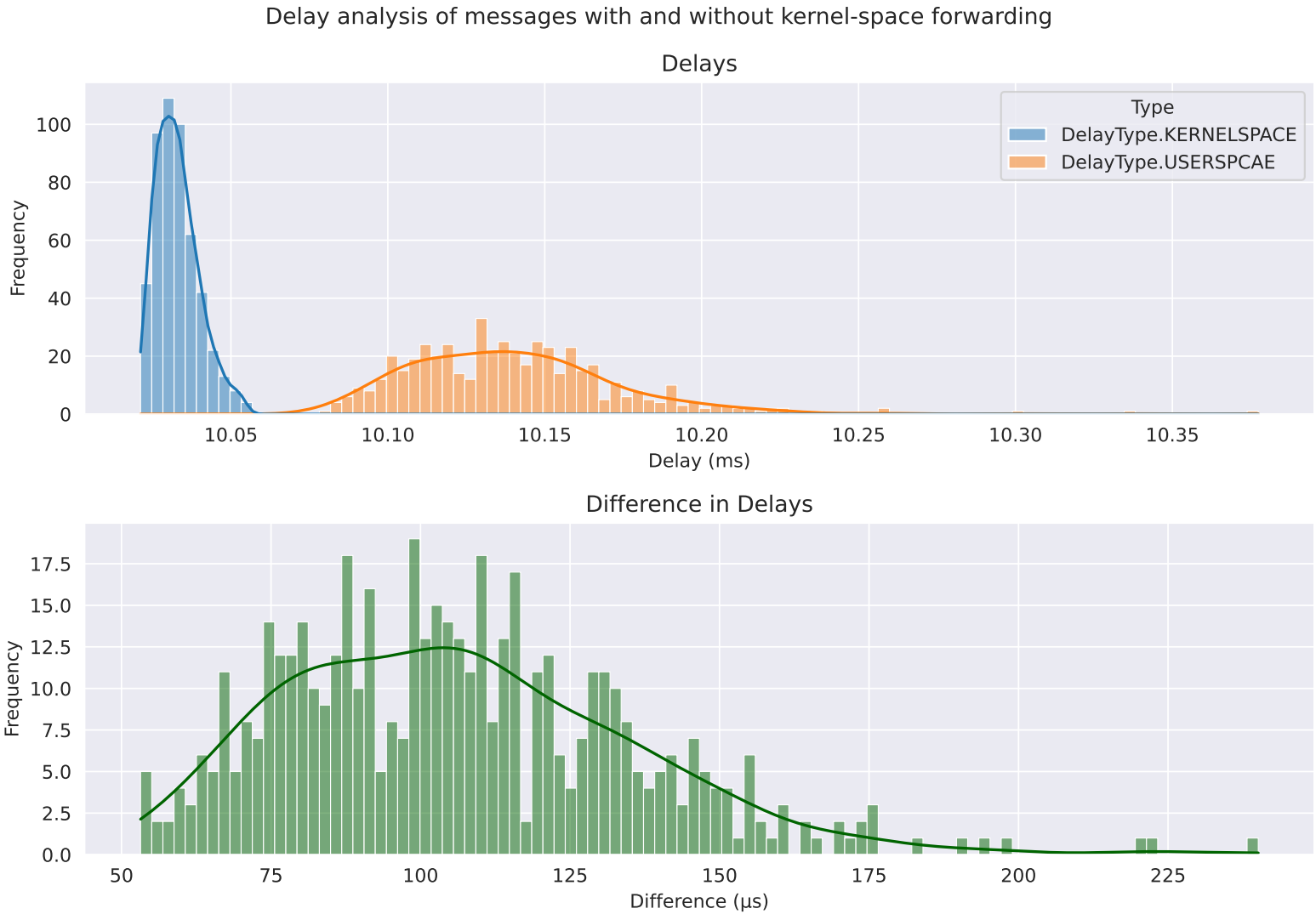
Packet Retransmission cont.



Test Setup

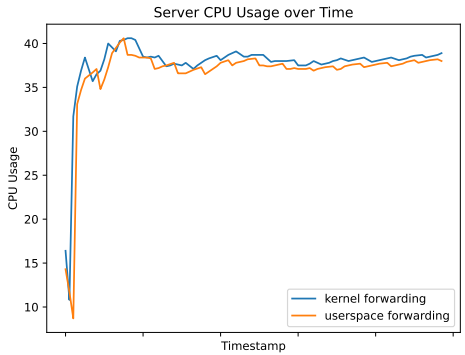
- Single machine setup (delay reduction only due to different kernel processing)
- Separate namespaces for client, relay, and server
- Artificial delay between client and relay for packet registration

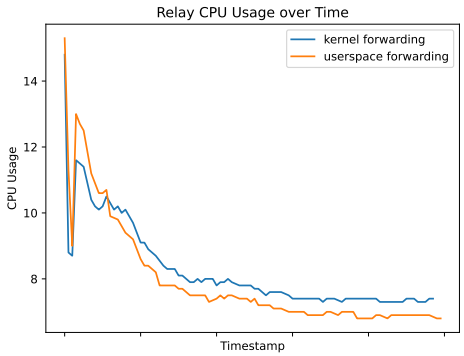
Test Results Delay Reduction

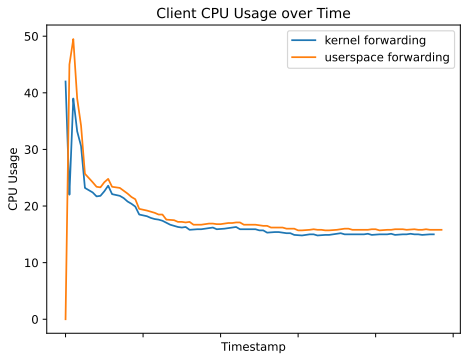


Test Results CPU Usage

- 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

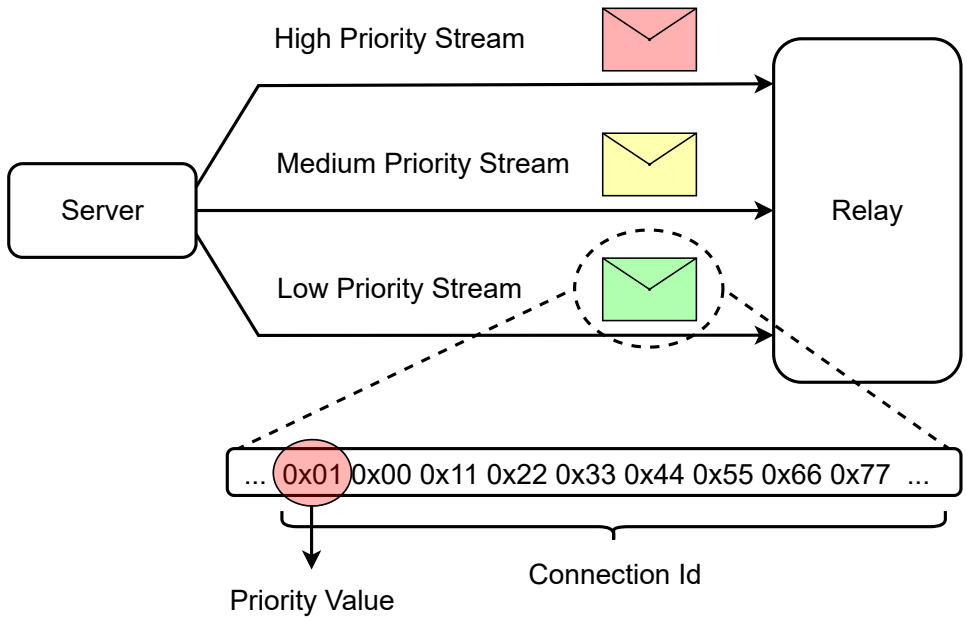
- Hardware offloading of en- and decryption
- Expand to other protocols
- Prototype completion
  - Congestion control
  - Physical setup for testing

That's it!

Any Questions?

Packet Priorities

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



Function Pointer Additions

```
1 /* Function pointer call within actual quic-go code */
2 if packet_setting.ConnectionIdUpdateBPFHandler != nil /* &&
      potentially other conditions */ {
3     packet_setting.ConnectionIdUpdateBPFHandler(connId.Bytes(),
          uint8(connId.Len()), p.connection)
4 }
```

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

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

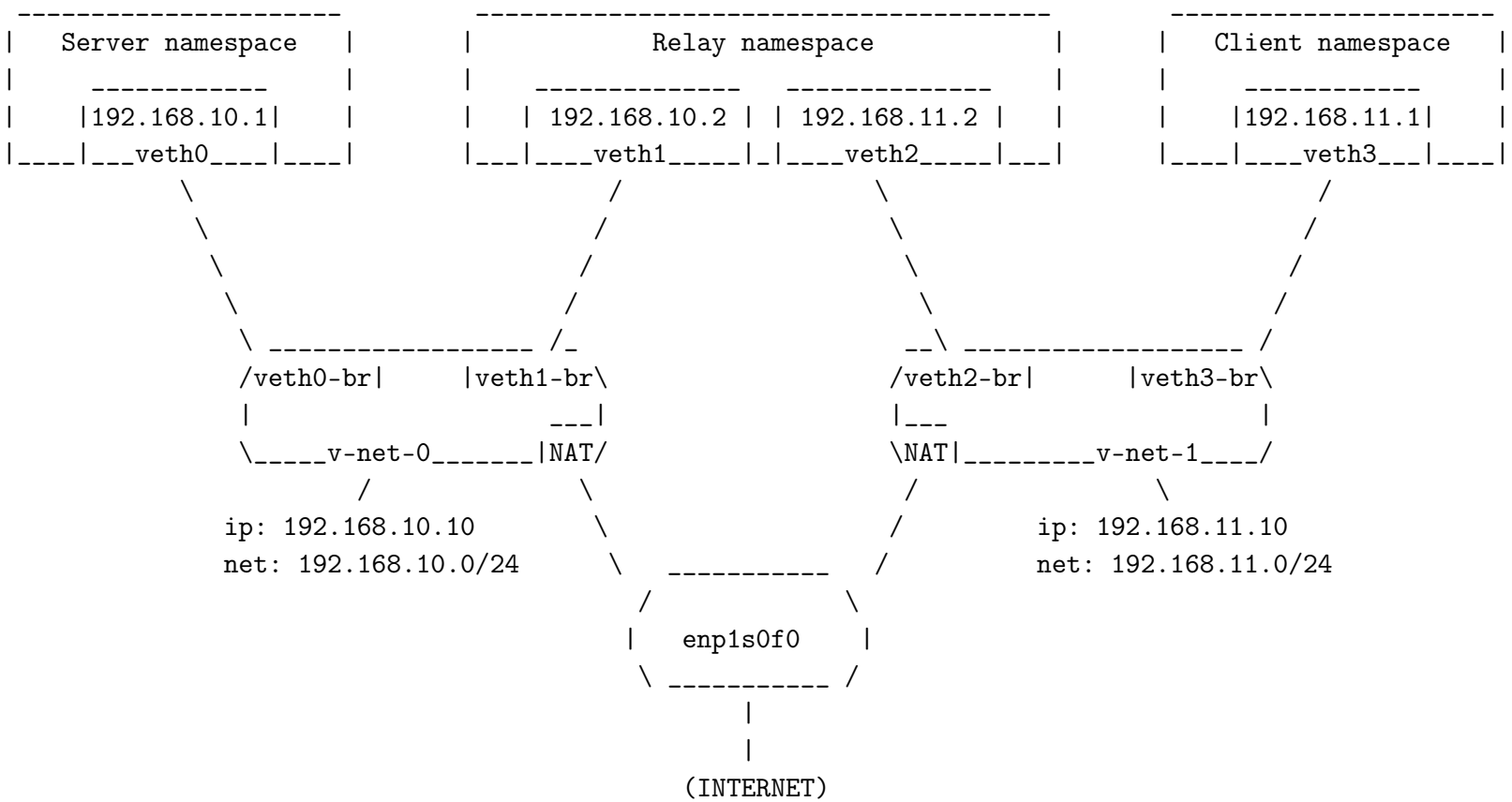
Listing 3: The signature will be defined within the library itself.

Function Pointer Additions

```
1  /* Definition of the function within the local relay code */
2  func localUpdateConnectionId(id []byte, l uint8, conn
       packet_setting.QuicConnection) {
3      /* handle the connection update by interacting with the eBPF
           program */
4  }
5
6  /* Providing the function to the quic-go library */
7  func main() {
8      /* ... */
9      packet_setting.ConnectionIdUpdateBPFHandler =
           localUpdateConnectionId
10     /* ... */
11 }
```

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

Test Setup


Server namespace
192.168.10.1
veth0
Relay namespace
192.168.10.2
192.168.11.2
veth1
veth2
Client namespace
192.168.11.1
veth3
/veth0-br
veth1-br\
v-net-0
NAT
/veth2-br
veth3-br\
NAT
v-net-1
ip: 192.168.10.10
net: 192.168.10.0/24
ip: 192.168.11.10
net: 192.168.11.0/24
enp1s0f0
(INTERNET)