Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

C++-based MASQUE-Proxying for Lower OSI-Layer Protocol Traffic

Final talk for the IDP by

Christoph Rotte

advised by Lion Steger and Richard von Seck

Monday 19th February, 2024

Chair of Network Architectures and Services
Department of Informatics
Technical University of Munich

Introduction

Motivation

- HTTP CONNECT: Method for creating TCP tunnels over HTTP
- **MASQUE** utilizes **HTTP/3** and **QUIC** for versatile **UDP/IP proxying**
- Early MASQUE stage with recent **CONNECT-UDP** and **CONNECT-IP** standardization
- First implementation efforts by Google's QUICHE[1]

[1] `https://github.com/google/quiche`

Introduction

Research Questions

1. Analyze the impact of **encapsulation overhead** on efficiency and operation
2. Evaluate transmission performance in terms of **transfer rates** and **reliability**
3. Investigate challenges affecting MASQUE's **implementation** and **functionality**

Background

QUIC and QUIC Datagrams

- QUIC: UDP-based TCP alternative (used for HTTP/3)
- Logical streams consist of **reliable** STREAM frames [1]
- **RFC9221**: Unreliable QUIC Datagram Extension [2]

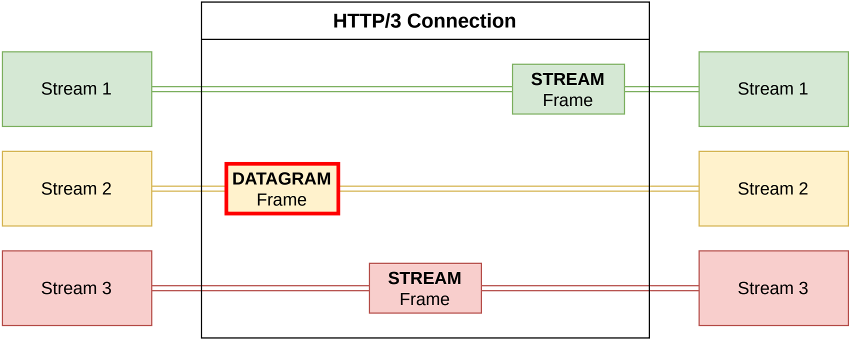

Figure 1: QUIC Streams Using Unreliable QUIC Datagrams

MASQUE-Proxying

CONNECT-UDP Method

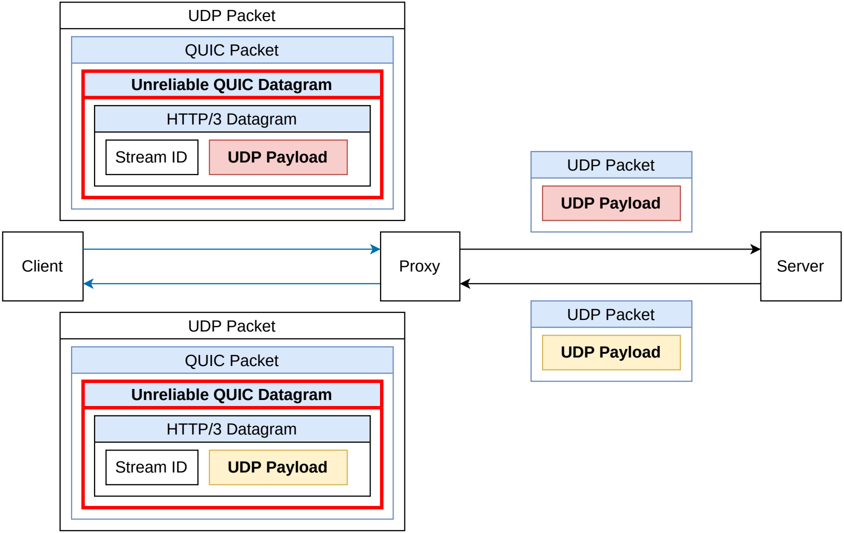

Figure 2: Proxying via QUIC and CONNECT-UDP [3]

MASQUE-Proxying

CONNECT-IP Method

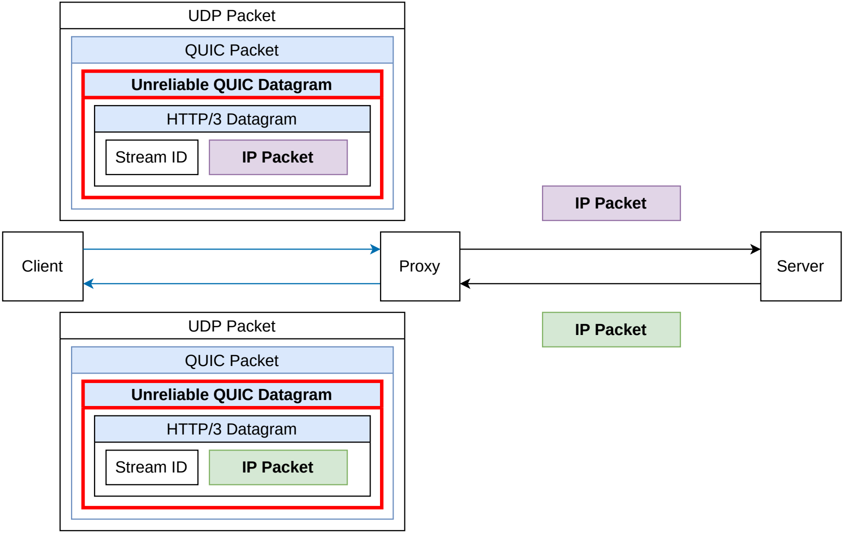

Figure 3: Proxying via QUIC and CONNECT-IP [4]

MASQUE-Proxying

Related Work

- Limited MASQUE research and implementations highlight early-stage development

- **Kühlewind et al.** offer a comprehensive analysis of MASQUE-based tunnel setup on end-to-end QUIC performance[2]
- **Scharnitzky et al.** developed an LTE emulation net device in ns-3[3] to analyze MASQUE proxying over emulated mobile networks[4]
- **iCloud Private Relay** uses MASQUE for user privacy [5]
- **Probst's work** (master's thesis) on MASQUE implementation, primarily focusing on privacy aspects and potential modifications for enhanced privacy [6]

[2] M. Kühlewind, M. Carlander-Reuterfelt, M. Ihlar, and M. Westerlund, Evaluation of quic-based masque proxying, in Proceedings of the 2021 Workshop on Evolution, Performance and Interoperability of QUIC, ser. EPIQ 21. New York, NY, USA: Association for Computing Machinery, 2021, p. 2934. [Online]. Available: `https://doi.org/10.1145/3488660.3493806`

[3] `https://www.nsnam.org/`

[4] D. Scharnitzky, Z. Krämer, S. Molnár, and A. Mihály, Real-time emulation of masque-based quic proxying in lte networks using ns-3.

Implementation

Design

- Limited related work prompts necessity for custom development
- Proxygen[5] and mvfst[6] by Facebook chosen for MASQUE implementation
  - Proxygen supports **HTTP/3 datagrams**, essential for MASQUE
  - Mvfst provides QUIC transport layer and **unreliable datagram support**
- Implementation leverages existing QUIC/HTTP functionalities for CONNECT-UDP/IP
- QUIC library comparison highlights mvfst's rich features and wide use

[5] `https://github.com/facebook/proxygen`
[6] `https://github.com/facebook/mvfst`

Implementation

CONNECT-IP Implementation

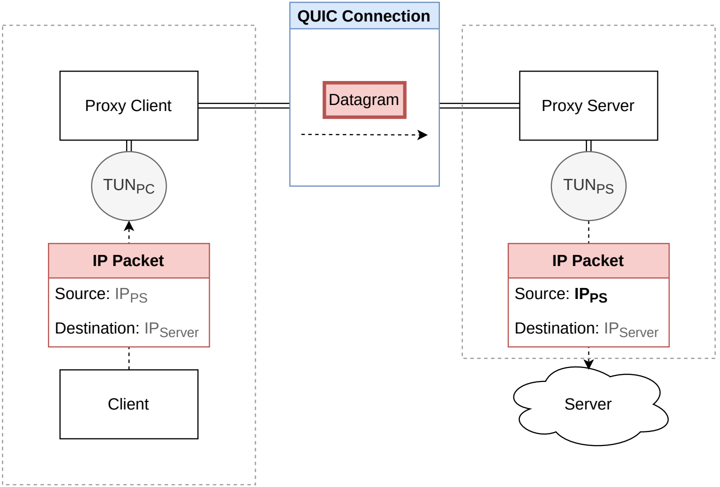

Figure 4: CONNECT-IP Implementation Overview

Implementation

Multiple Hops

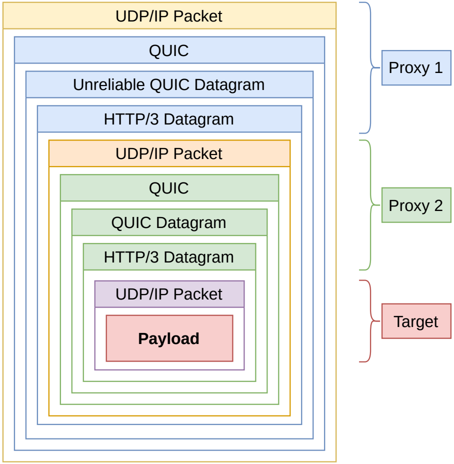

Figure 5: Layered CONNECT-IP Packet

- For each proxy hop, we have to add HTTP/3 + QUIC + UDP/IP headers
- **Default MTU**: 1500B
- **Minimum QUIC MTU**: 1280B

→ Practical limit for the number of encapsulated layers (3 - 4 hops)

Evaluation

Setup

- **Testbed**: Nine servers[7] with 1Gbit/s on `eno5` and 10Gbit/s via `eno3` and `eno4`
- **Parameters**: # Hops, # Clients, # Streams / Transactions
- **Metrics**: Throughput, TTFB (QUIC / HTTP), RTT (QUIC), Latency (QUIC), Jitter (QUIC)

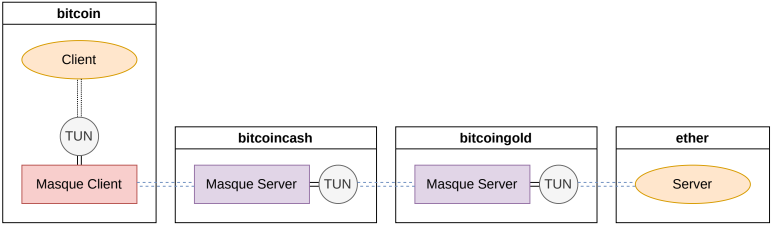

Figure 6: Two-Hop Sequence

[7] OS: Debian 10 | CPU: Intel Xeon D-1518 (4 cores / 8 threads) | Memory: 32GB
[8] 10% packet loss | 200ms RTT | 1Mbit/s bandwidth

Evaluation

Setup

- **Testbed**: Nine servers[7] with 1Gbit/s on `eno5` and 10Gbit/s via `eno3` and `eno4`
- **Parameters**: # Hops, # Clients, # Streams / Transactions
- **Metrics**: Throughput, TTFB (QUIC / HTTP), RTT (QUIC), Latency (QUIC), Jitter (QUIC)

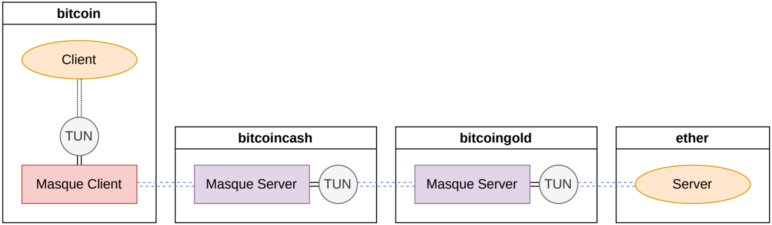

Figure 6: Two-Hop Sequence

- **Setups and Motivation**:
  - I **TunConnectIP**: General usage with external applications (`iperf`) ($\leq$ 6 servers)
  - II **HTTPConnectIP + HTTPConnectUDP**: Minimized client overhead (GET of 10GB) ($\leq$ 6 servers)
  - III **SeleniumConnectIP**: User experience w/ realistic environment[8] (GET `tum.de`) ($\leq$ 9 servers)

[7] OS: Debian 10 | CPU: Intel Xeon D-1518 (4 cores / 8 threads) | Memory: 32GB
[8] 10% packet loss | 200ms RTT | 1Mbit/s bandwidth

Evaluation

I TunConnectIP Throughput

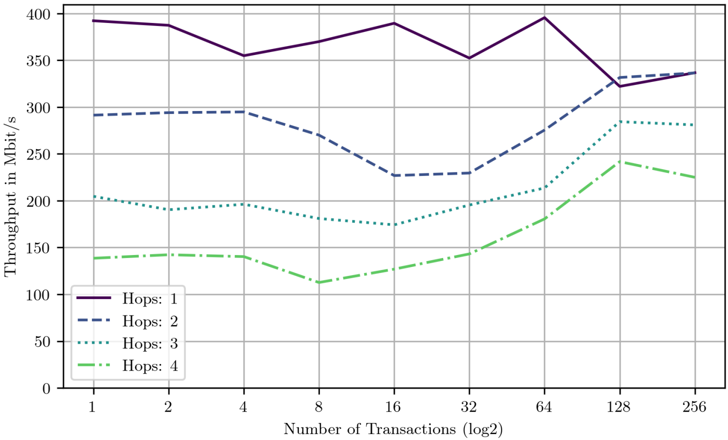

Figure 7: TUNConnectIP Throughput

→ Throughput declines with increased hops → Stabilizes after initial transactions

Evaluation

II HTTPConnectIP Throughput

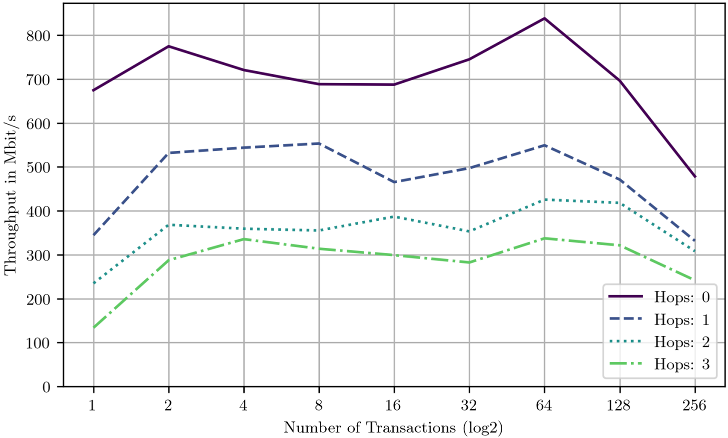

Figure 8: HTTPConnectIP Throughput

→ Higher throughput than TunConnectIP

→ T=64: Scheduling behavior

Evaluation

II HTTPConnectIP RTT

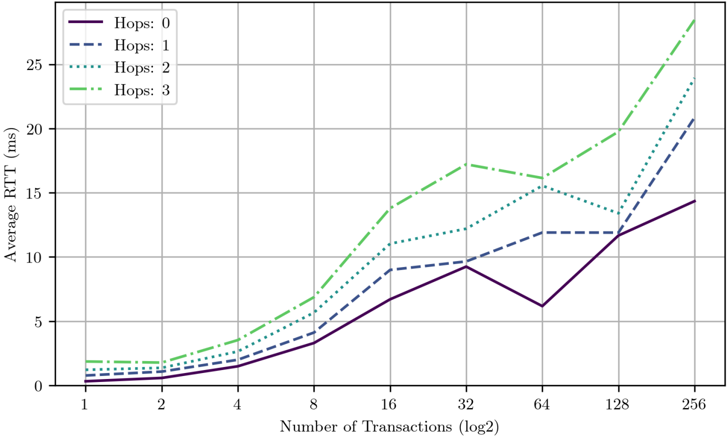

Figure 9: HTTPConnectIP RTT

→ Consistent upward trend for all hops

→ Fluctuations: Underlying library dynamics

Evaluation

II HTTPConnectIP Latency

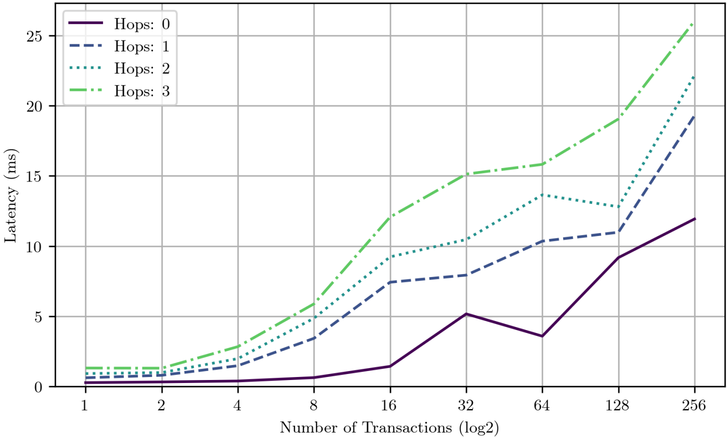

Figure 10: HTTPConnectIP Latency

→ Latency trends align with RTT

→ Fluctuations: Underlying library dynamics

Evaluation

II HTTPConnectIP Jitter

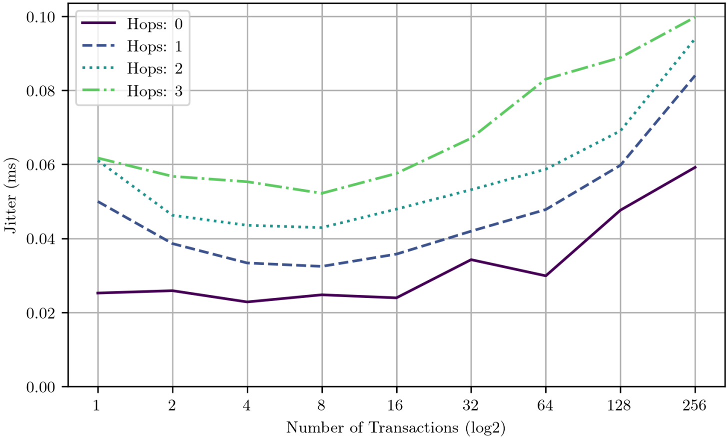

Figure 11: HTTPConnectIP Jitter

→ Patterns correlate with CPU utilization

→ Observable variations with hop count

Evaluation

III SeleniumConnectIP

- **Page Load Time**:
  - Increases with more transactions/clients and hops
  - Higher hops accelerate load time and variance increase
    - 6 clients / **0 hops** / 16 transactions: Main range between **2s** and **5s**
    - 6 clients / **2 hops** / 16 transactions: Main range between **3s** and **190s**

Evaluation

III SeleniumConnectIP

- **Page Load Time**:
  - Increases with more transactions/clients and hops
  - Higher hops accelerate load time and variance increase
    - 6 clients / **0 hops** / 16 transactions: Main range between **2s** and **5s**
    - 6 clients / **2 hops** / 16 transactions: Main range between **3s** and **190s**
- **TTFB Stability**:
  - Consistent across different transaction/client counts
    - 1 client / **0 hops** / 1 transaction: Median of ≈**400ms**
    - 6 clients / **2 hops** / 16 transactions: Median of ≈**450ms**
  - Lower increase in variance than load time

Evaluation

III SeleniumConnectIP

- **Page Load Time**:
  - Increases with more transactions/clients and hops
  - Higher hops accelerate load time and variance increase
    - 6 clients / **0 hops** / 16 transactions: Main range between **2s** and **5s**
    - 6 clients / **2 hops** / 16 transactions: Main range between **3s** and **190s**
- **TTFB Stability**:
  - Consistent across different transaction/client counts
    - 1 client / **0 hops** / 1 transaction: Median of $\approx$**400ms**
    - 6 clients / **2 hops** / 16 transactions: Median of $\approx$**450ms**
  - Lower increase in variance than load time
- **Overall Conclusion**:
  - Page loading heavily influenced by proxy hops and complexity
    - Unknown factors of Chrome
  - TTFB relatively stable, less affected by proxy setup

Conclusion

- **Encapsulation overhead** and **Transmission performance**:
  - Expected performance impacts with multiple hops
    - HTTPConnectIP (2 Transactions): 1 Hop: 530 Mbit/s | 2 Hops: 370 Mbit/s
  - Scales with multiple parallel transactions
    - HTTPConnectIP (3 Hops): 1 Transaction: 140 Mbit/s | 64 Transactions: 340 Mbit/s
  - Stable initial TTFB despite proxying complexity

Conclusion

- **Encapsulation overhead** and **Transmission performance**:
  - Expected performance impacts with multiple hops
    - HTTPConnectIP (2 Transactions): 1 Hop: 530 Mbit/s | 2 Hops: 370 Mbit/s
  - Scales with multiple parallel transactions
    - HTTPConnectIP (3 Hops): 1 Transaction: 140 Mbit/s | 64 Transactions: 340 Mbit/s
  - Stable initial TTFB despite proxying complexity
- **Implementation**:
  - Significant modifications required within libraries
  - Cross-testing difficult due to early stage

Conclusion

- **Encapsulation overhead** and **Transmission performance**:
  - Expected performance impacts with multiple hops
    - HTTPConnectIP (2 Transactions): 1 Hop: 530 Mbit/s | 2 Hops: 370 Mbit/s
  - Scales with multiple parallel transactions
    - HTTPConnectIP (3 Hops): 1 Transaction: 140 Mbit/s | 64 Transactions: 340 Mbit/s
  - Stable initial TTFB despite proxying complexity
- **Implementation**:
  - Significant modifications required within libraries
  - Cross-testing difficult due to early stage
- **Future work**:
  - Exploring alternative libraries for MASQUE implementation
  - Enhancing testbed realism for more accurate performance evaluation
  - Comparing MASQUE with other proxy protocols to identify potential improvements

Bibliography

[1] J. Iyengar and M. Thomson, “QUIC: A UDP-Based Multiplexed and Secure Transport,” 2021, accessed: 20.05.2023. [Online]. Available: http://tools.ietf.org/html/rfc9000

[2] T. Pauly, E. Kinnear, and D. Schinazi, “An Unreliable Datagram Extension to QUIC,” 2022, accessed: 20.05.2023. [Online]. Available: http://tools.ietf.org/html/rfc9221

[3] D. Schinazi, “Proxying UDP in HTTP,” 2022, accessed: 20.05.2023. [Online]. Available: http://tools.ietf.org/html/rfc9298

[4] T. Pauly, D. Schinazi, A. Chernyakhovsky, M. Kühlewind, and M. Westerlund, “Proxying IP in HTTP,” 2023, accessed: 20.05.2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9484

[5] Apple, “icloud private relay overview,” 2021, accessed: 20.05.2023. [Online]. Available: https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf

[6] C. Probst, “Rust-based MASQUE-Proxying for Lower OSI-Layer Protocol Traffic,” MA, 2022, Lion Steger, Richard von Seck.

Appendix

II HTTPConnectIP TTFB

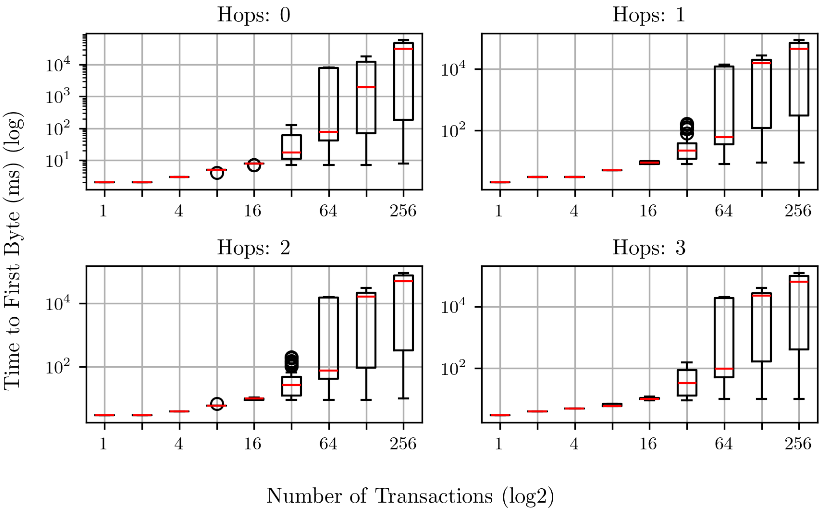

Figure 12: HTTPConnectIP TTFB

Appendix

III SeleniumConnectIP Page Load Times

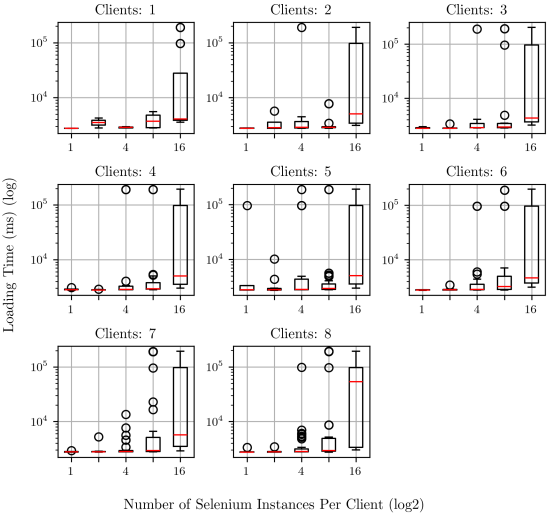

Figure 13: SeleniumConnectIP Page Load Times (0 Hops)

Appendix

III SeleniumConnectIP Page Load Times

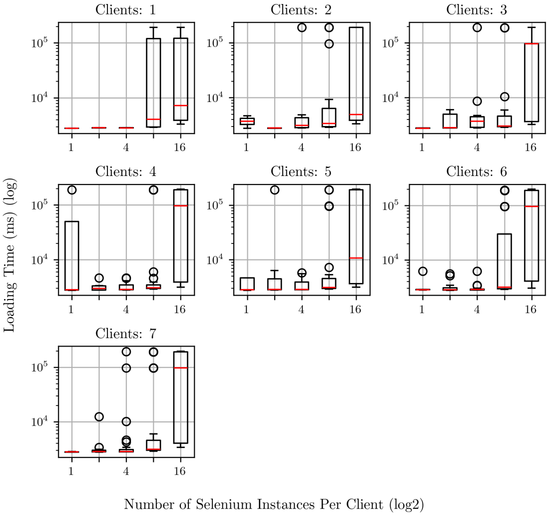

Figure 14: SeleniumConnectIP Page Load Times (1 Hop)

III SeleniumConnectIP Page Load Times

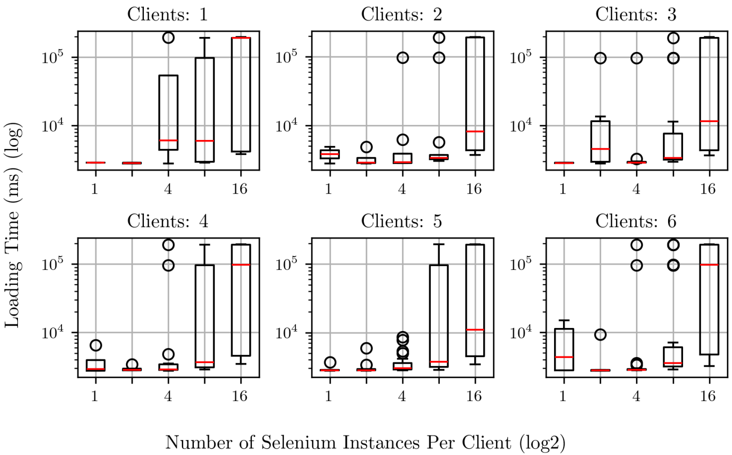

Figure 15: SeleniumConnectIP Page Load Times (2 Hops)

Appendix

III SeleniumConnectIP TTFB

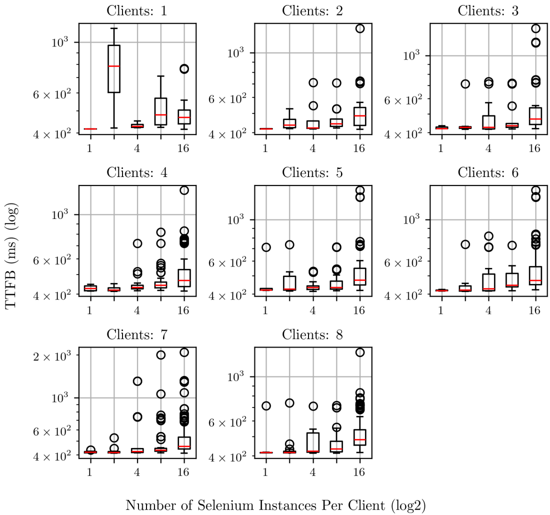

Figure 16: SeleniumConnectIP HTTP TTFB (0 Hops)

Appendix

III SeleniumConnectIP TTFB

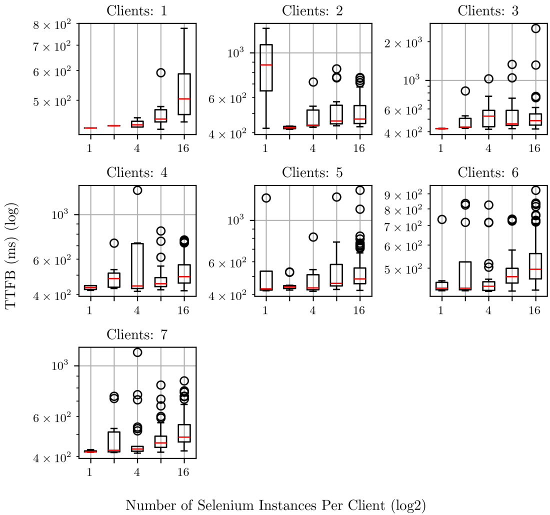

Figure 17: SeleniumConnectIP HTTP TTFB (1 Hop)

III SeleniumConnectIP TTFB

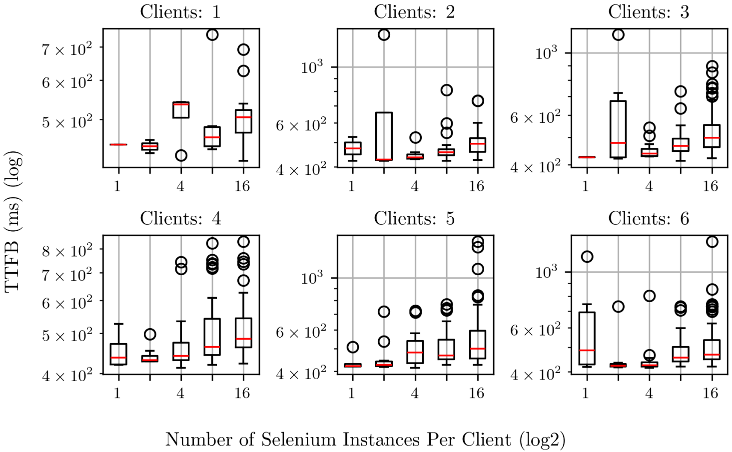

Figure 18: SeleniumConnectIP HTTP TTFB (2 Hops)