Cascades of Nested Acknowledgments in Multi-Hop MASQUE

ANRW 2025 Paper Presentation

Kathrin Elmenhorst – kelmenhorst@uos.de
Mirja Kühlewind, Ike Kunze, Constantin Sander, Klaus Wehrle

SYS | UNIVERSITÄT OSNABRÜCK
COM SYS | RWTH AACHEN UNIVERSITY

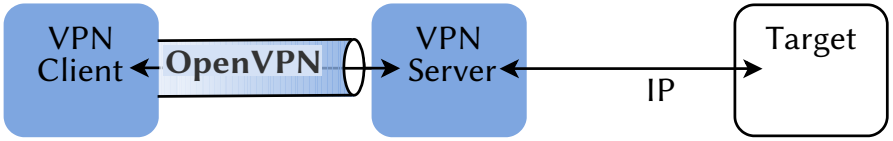
Motivation: QUIC Tunnels for IP Privacy

Motivation: QUIC Tunnels for IP Privacy

- Threats to user privacy on the Internet: IP can be used as trackable identifier

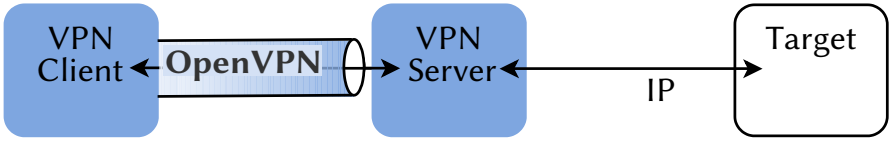
Motivation: QUIC Tunnels for IP Privacy

- Threats to user privacy on the Internet: IP can be used as trackable identifier
- **Proxying and tunneling** to hide endpoint IP addresses
  - e.g., VPN, Tor



Motivation: QUIC Tunnels for IP Privacy

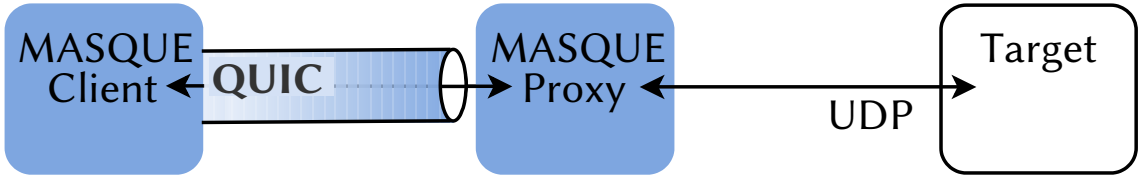
- Threats to user privacy on the Internet: IP can be used as trackable identifier
- **Proxying and tunneling** to hide endpoint IP addresses
  - e.g., VPN, Tor



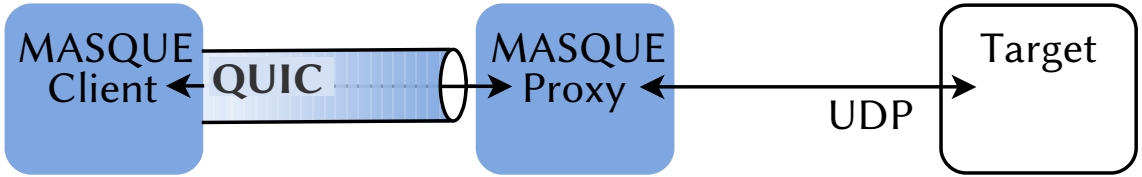
- Today: Transport protocol **QUIC provides advantages for tunneling**
  - Built-in encryption
  - Faster connection setup
  - Stream-based multiplexing
  - Based on UDP ➤ Optional unreliable congestion-controlled transport

Kathrin Elmenhorst

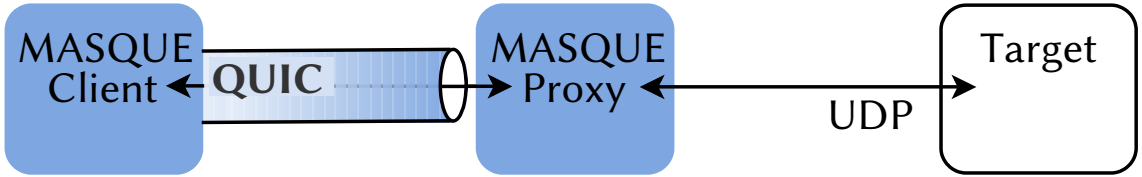
Background: QUIC-based MASQUE – Tunneling UDP through QUIC



Background: QUIC-based MASQUE – Tunneling UDP through QUIC

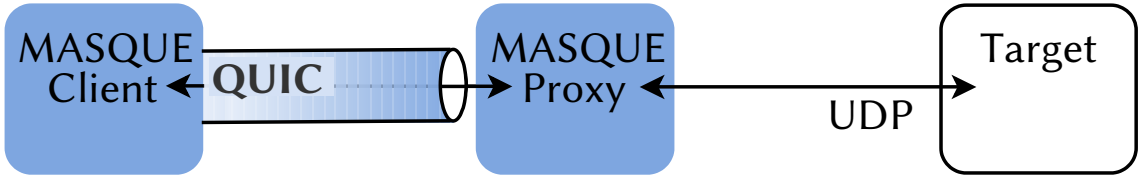


Background: QUIC-based MASQUE – Tunneling UDP through QUIC



- UDP traffic tunneled via QUIC connection between MASQUE client and MASQUE proxy, then forwarded to target

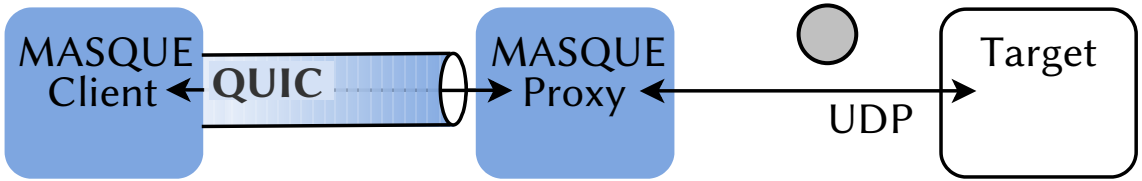
Background: QUIC-based MASQUE – Tunneling UDP through QUIC



- UDP traffic tunneled via QUIC connection between MASQUE client and MASQUE proxy, then forwarded to target

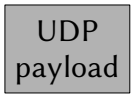
- **Tunnel = unreliable QUIC connection** provides encapsulation and encryption

Background: QUIC-based MASQUE – Tunneling UDP through QUIC



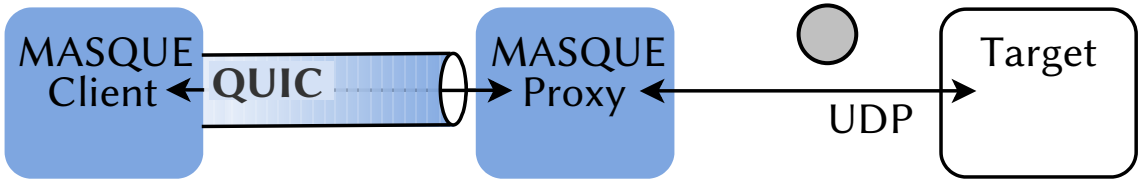
- UDP traffic tunneled via QUIC connection between MASQUE client and MASQUE proxy, then forwarded to target

- **Tunnel = unreliable QUIC connection** provides encapsulation and encryption
  - Tunneled UDP payload



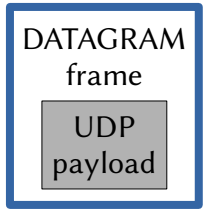
MASQUE encapsulation

Background: QUIC-based MASQUE – Tunneling UDP through QUIC



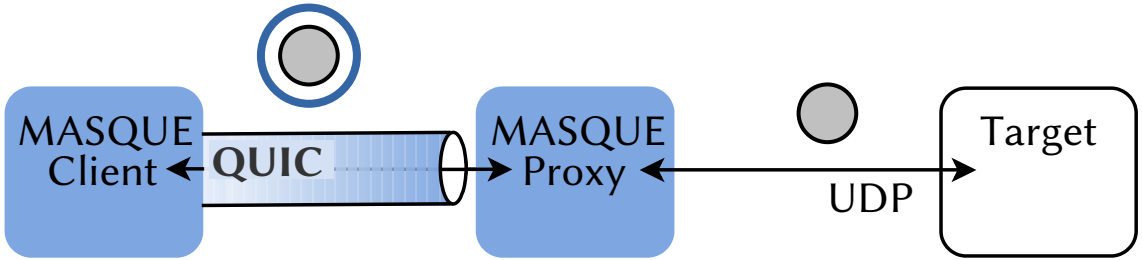
- UDP traffic tunneled via QUIC connection between MASQUE client and MASQUE proxy, then forwarded to target

- **Tunnel = unreliable QUIC connection** provides encapsulation and encryption
  - Tunneled UDP payload ➤ DATAGRAM frame ➤ encrypted



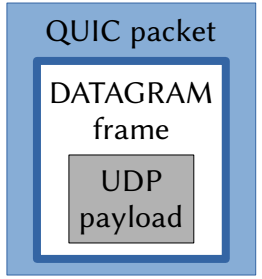
MASQUE encapsulation

Background: QUIC-based MASQUE – Tunneling UDP through QUIC



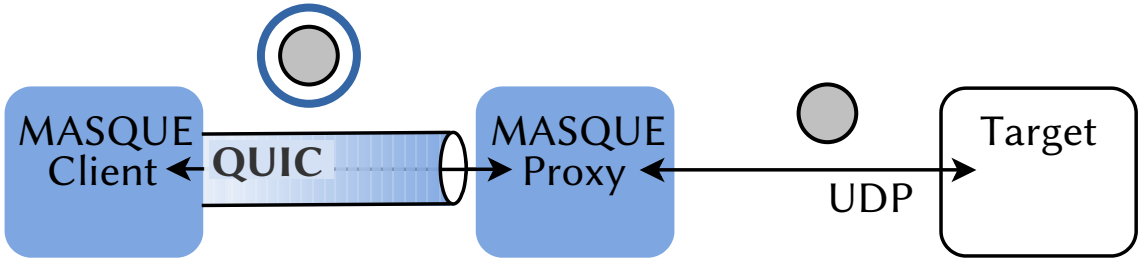
- UDP traffic tunneled via QUIC connection between MASQUE client and MASQUE proxy, then forwarded to target

- **Tunnel = unreliable QUIC connection** provides encapsulation and encryption
  - Tunneled UDP payload ➤ DATAGRAM frame ➤ encrypted ➤ QUIC packet



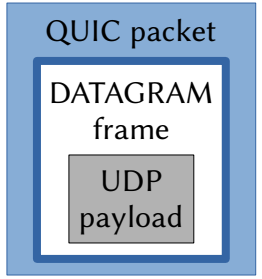
MASQUE encapsulation

Background: QUIC-based MASQUE – Tunneling UDP through QUIC



- UDP traffic tunneled via QUIC connection between MASQUE client and MASQUE proxy, then forwarded to target

- **Tunnel = unreliable QUIC connection** provides encapsulation and encryption
  - Tunneled UDP payload ➤ DATAGRAM frame ➤ encrypted ➤ QUIC packet
  - DATAGRAM: Congestion controlled ⟶ ACK-eliciting

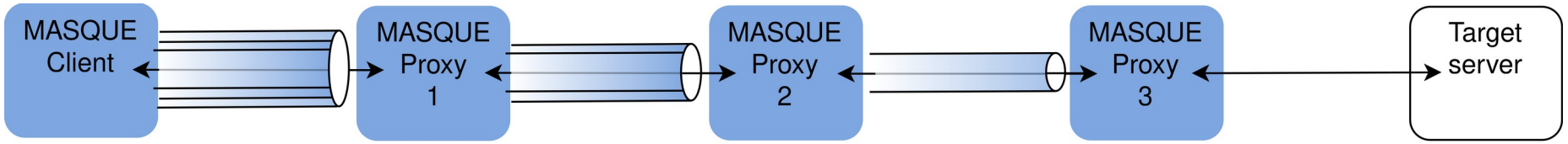


MASQUE encapsulation

Multi-Hop MASQUE

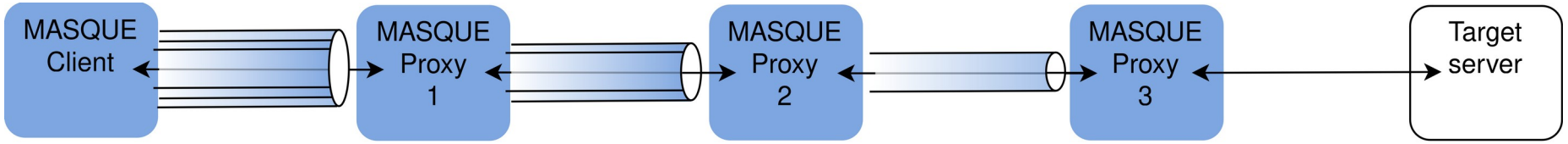
Multi-Hop MASQUE

- Client negotiates chain of proxies ⟶ nested tunnels



Multi-Hop MASQUE

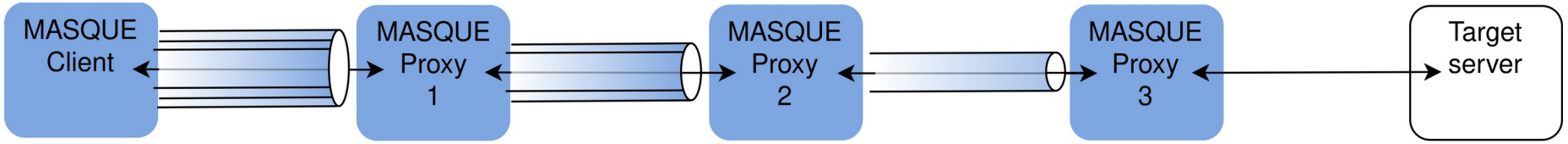
- Client negotiates chain of proxies ⟶ nested tunnels



- Apple feature “Private Relay” proxies all Safari web traffic through **2 MASQUE proxies**
  - Several privacy and performance aspects studied by previous work [Trevisan et al., 2022; Zohaib et al., 2023]

Multi-Hop MASQUE

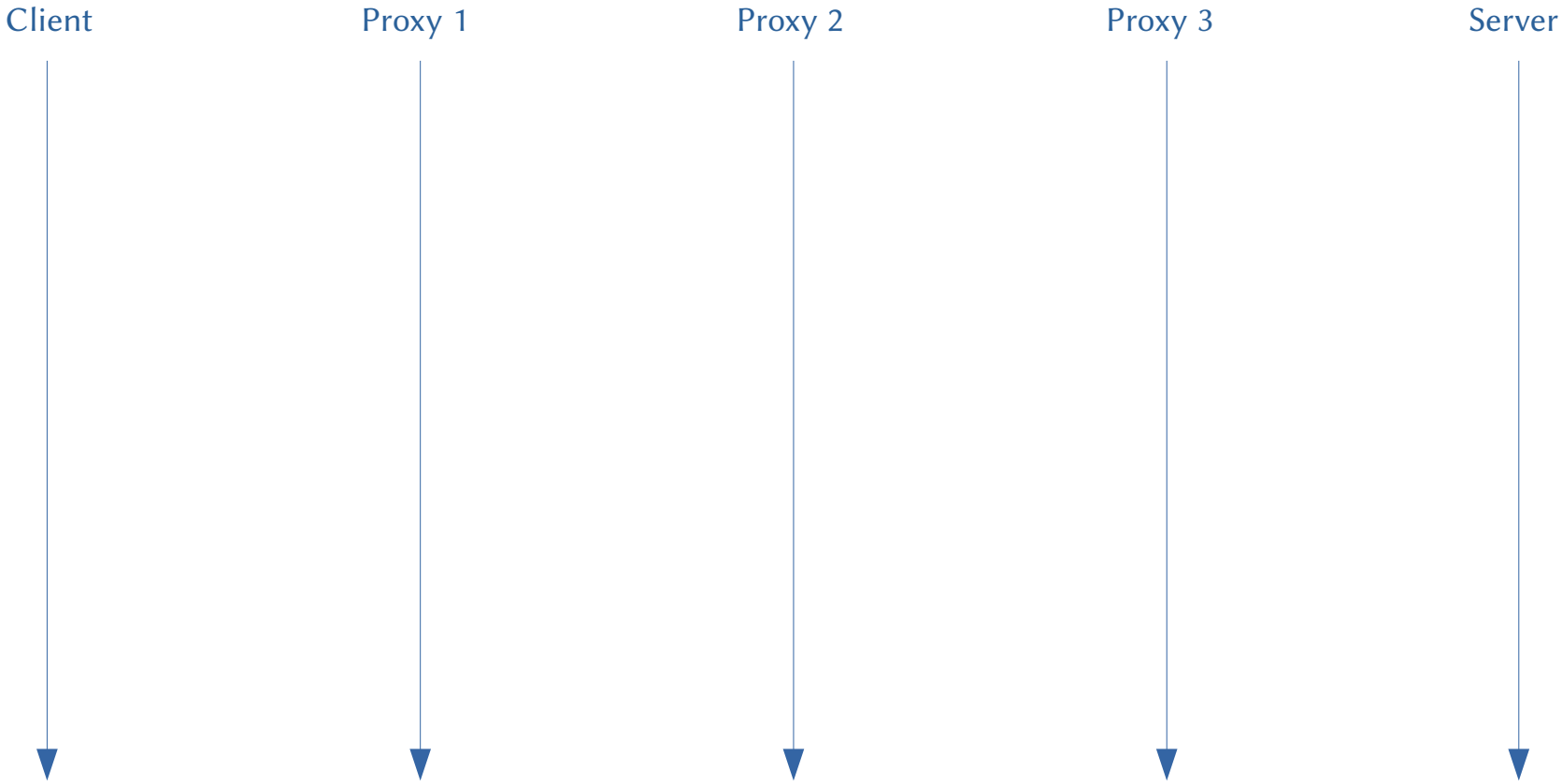
- Client negotiates chain of proxies ⟶ nested tunnels



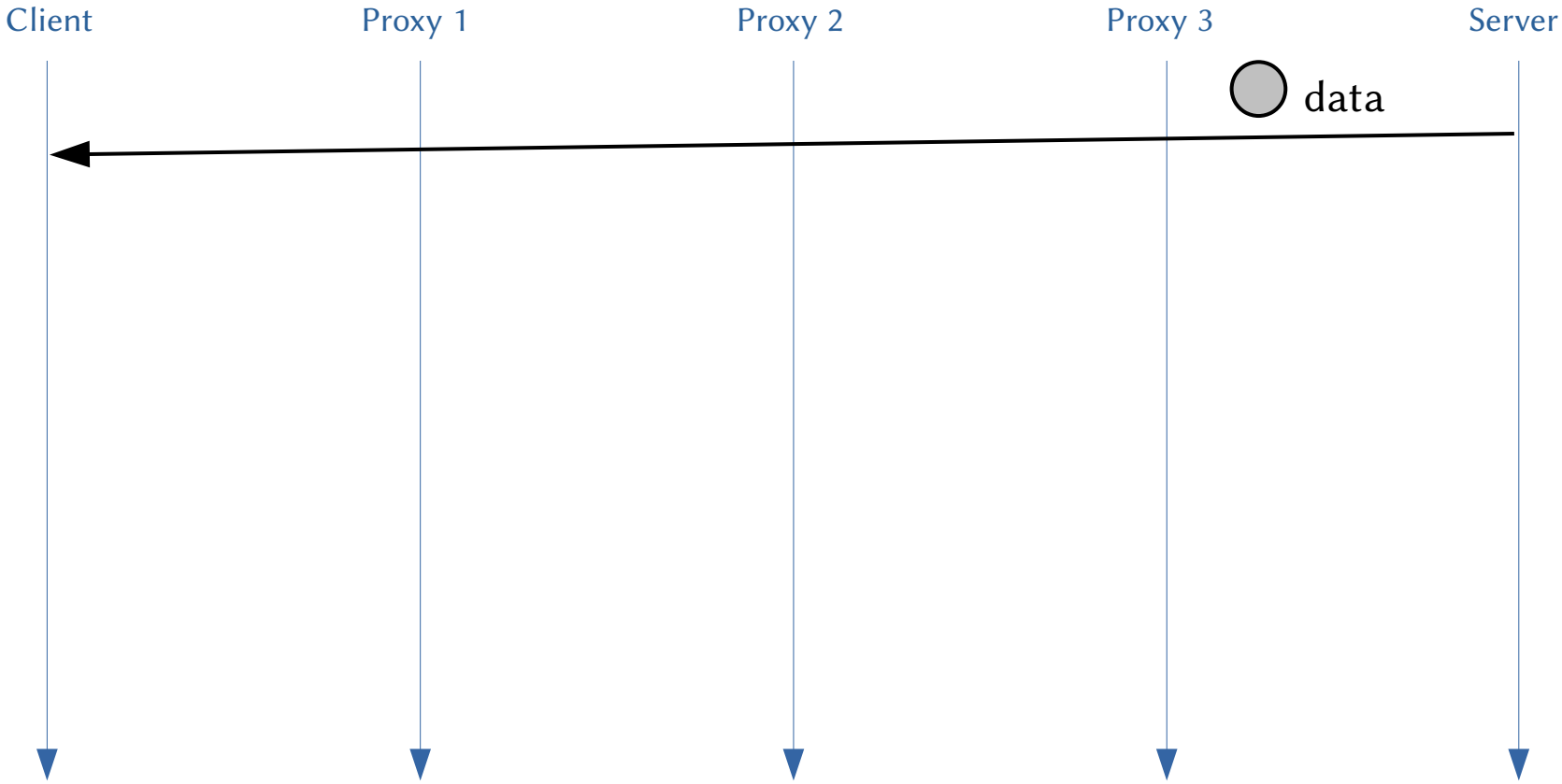
- Apple feature "Private Relay" proxies all Safari web traffic through **2 MASQUE proxies**
  - Several privacy and performance aspects studied by previous work [Trevisan et al., 2022; Zohaib et al., 2023]

- Our investigation:
  - **Generic** multi-hop MASQUE performance independent of Apple's infrastructure, and
  - **With more than 2 proxies**, e.g., to reduce collusion-risk for a stricter privacy use case

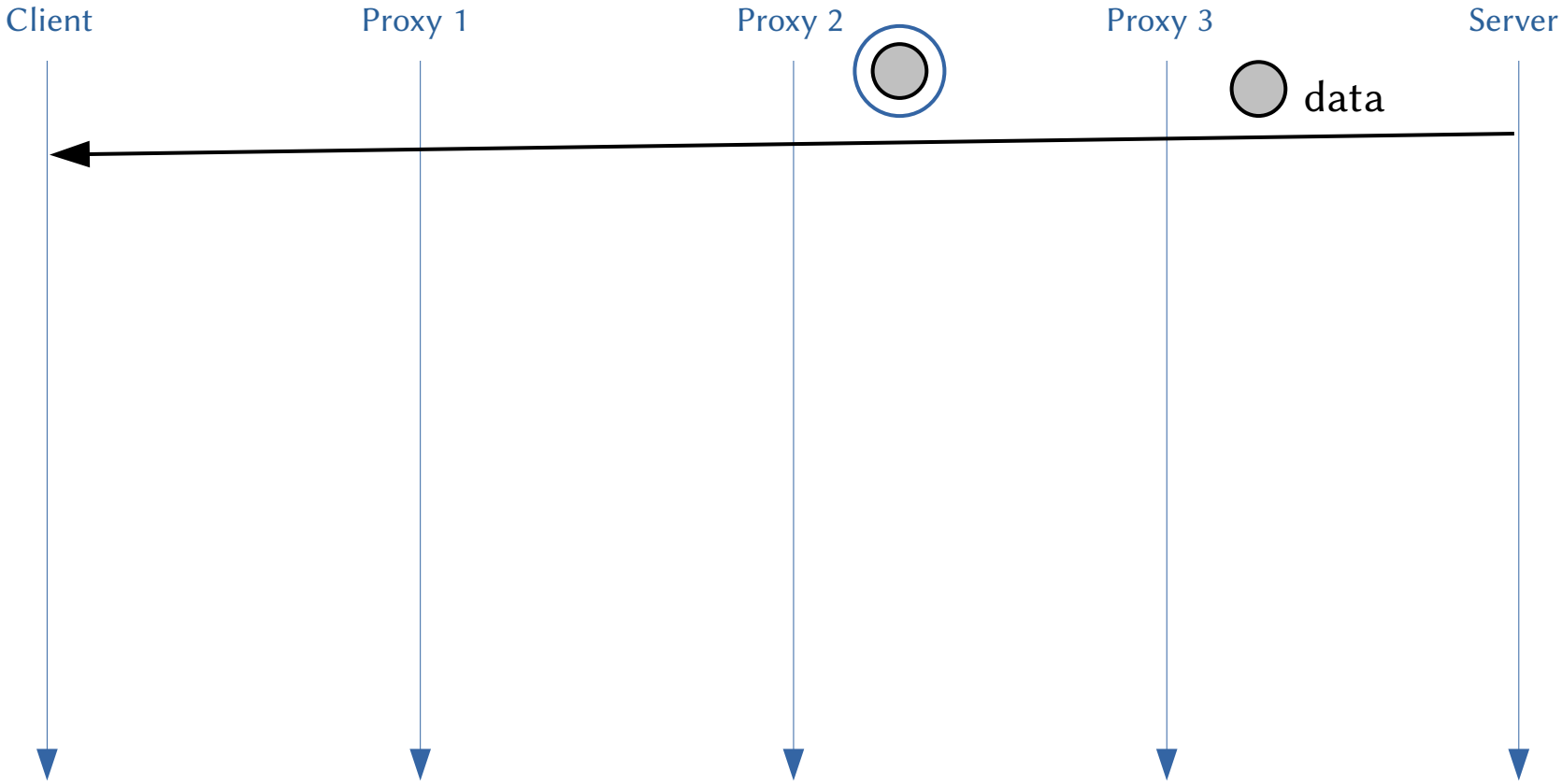
Analysis: Nested ACKs



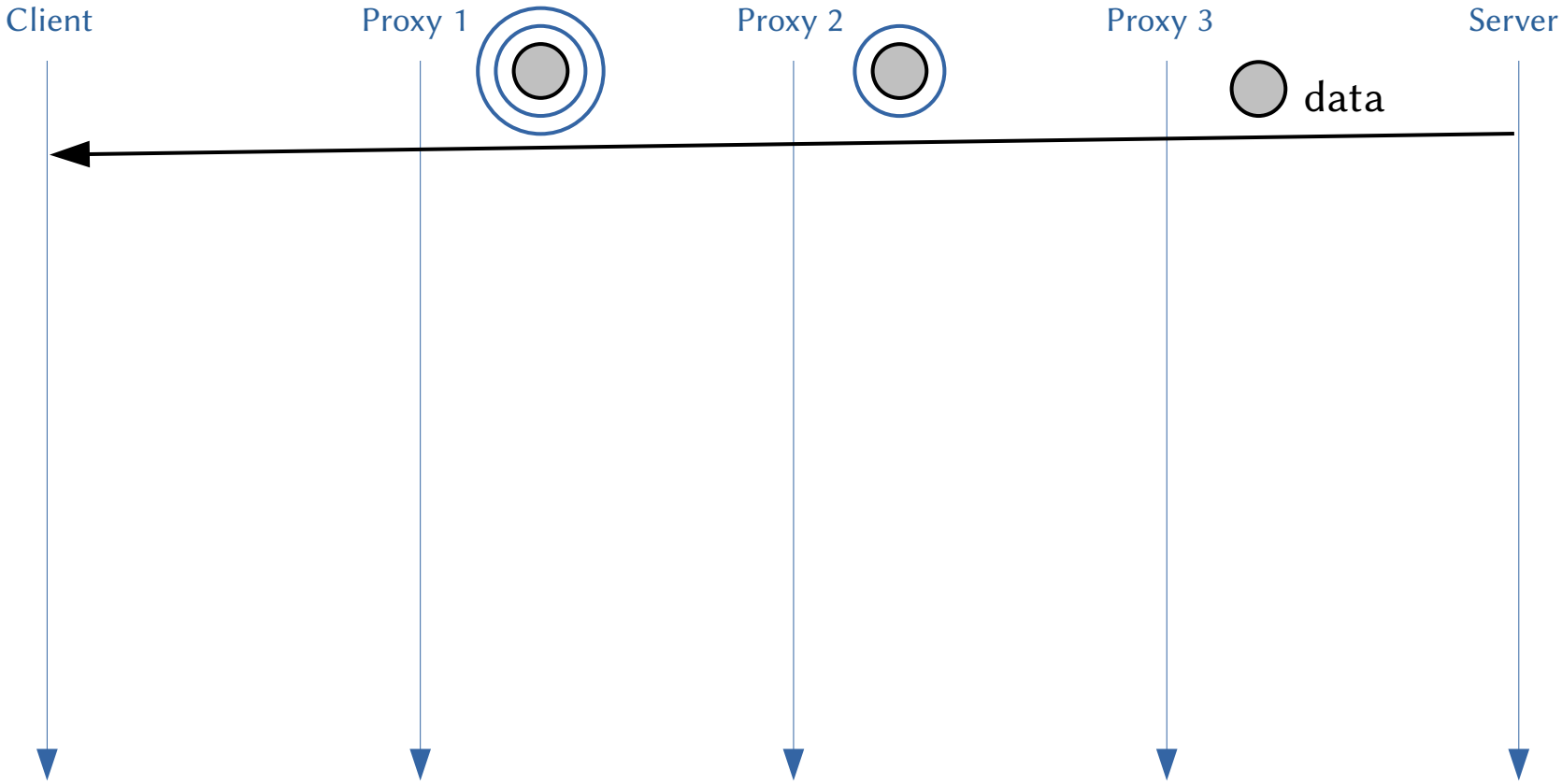
Analysis: Nested ACKs



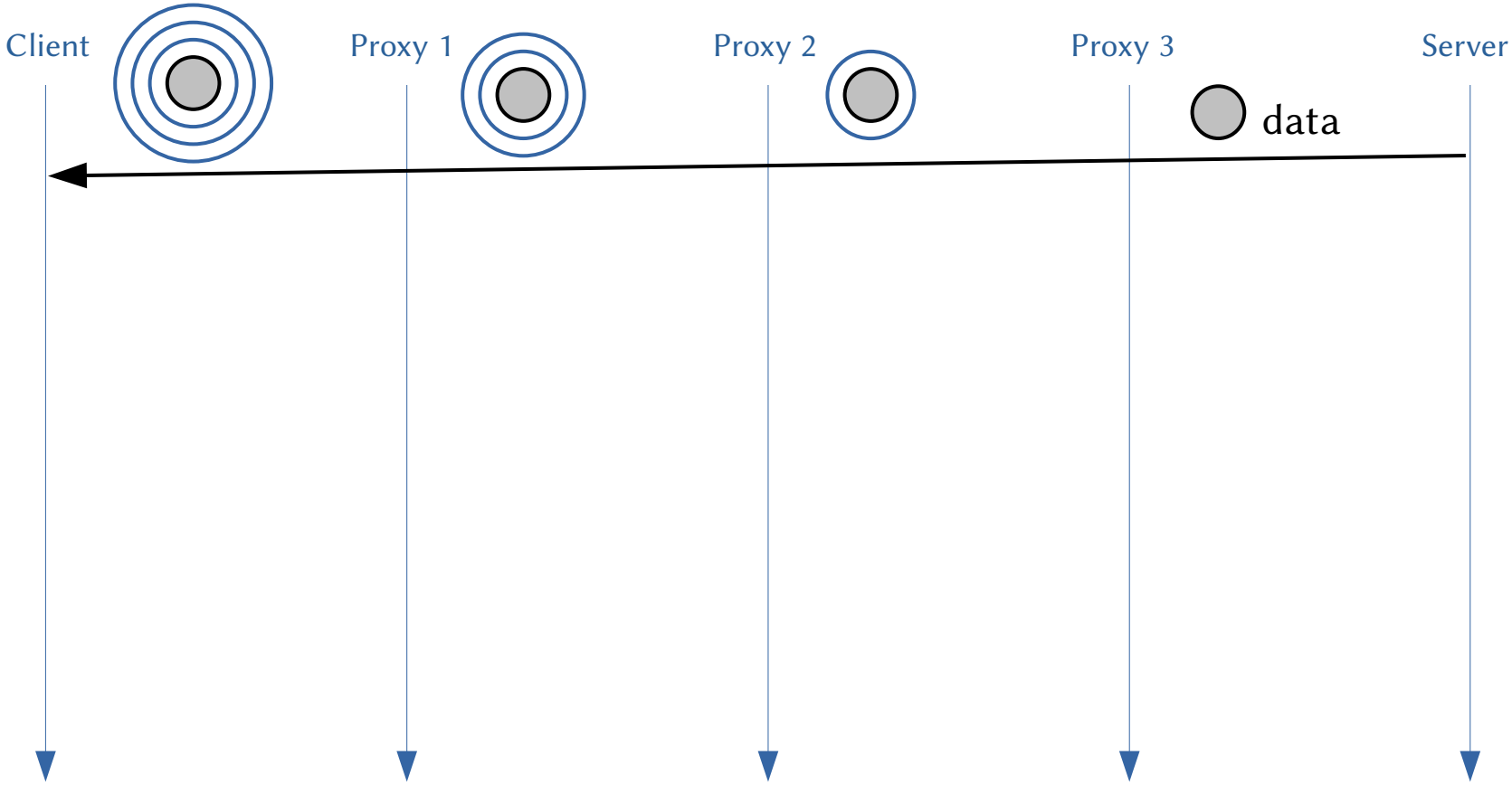
Analysis: Nested ACKs



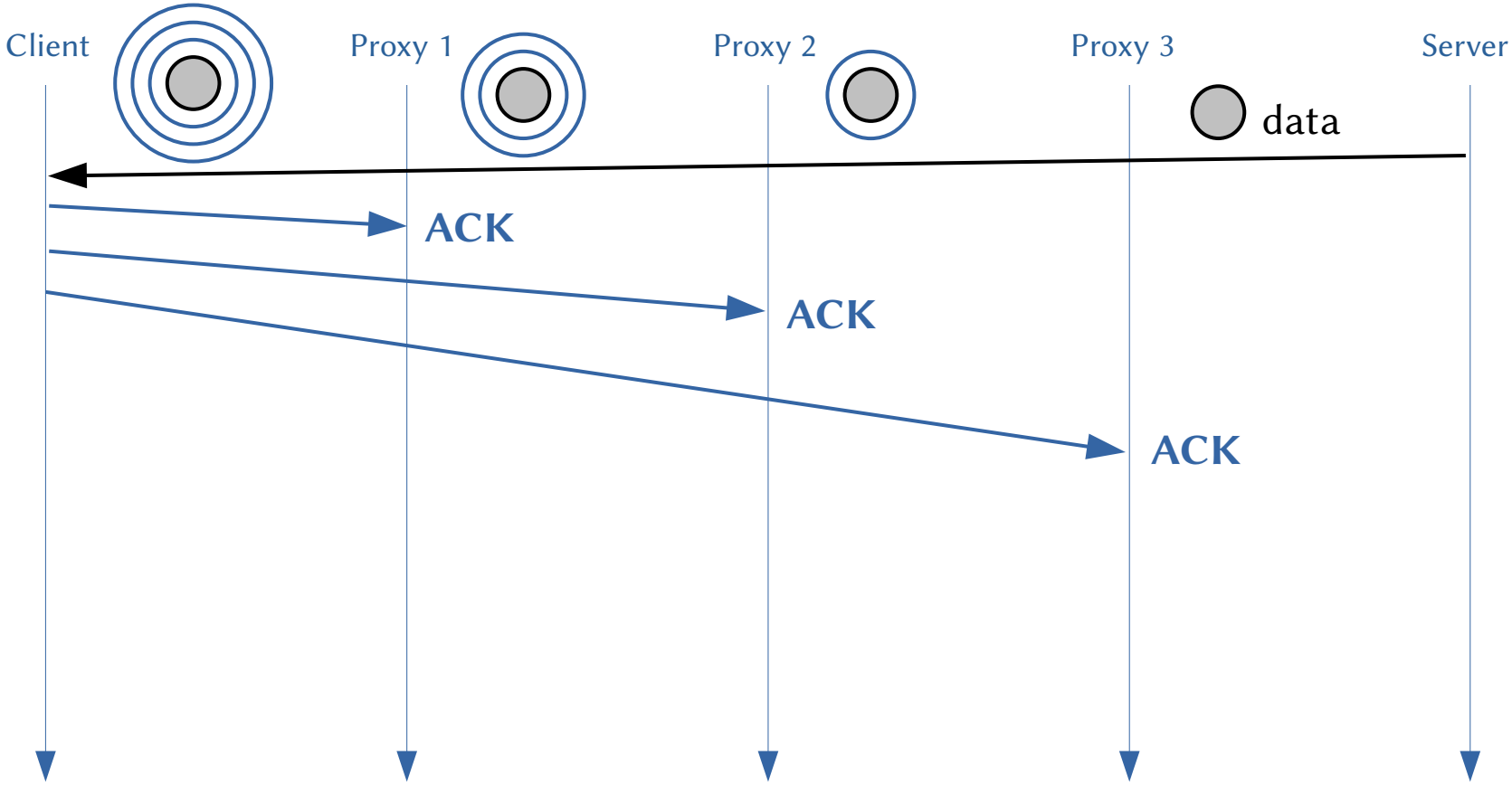
Analysis: Nested ACKs



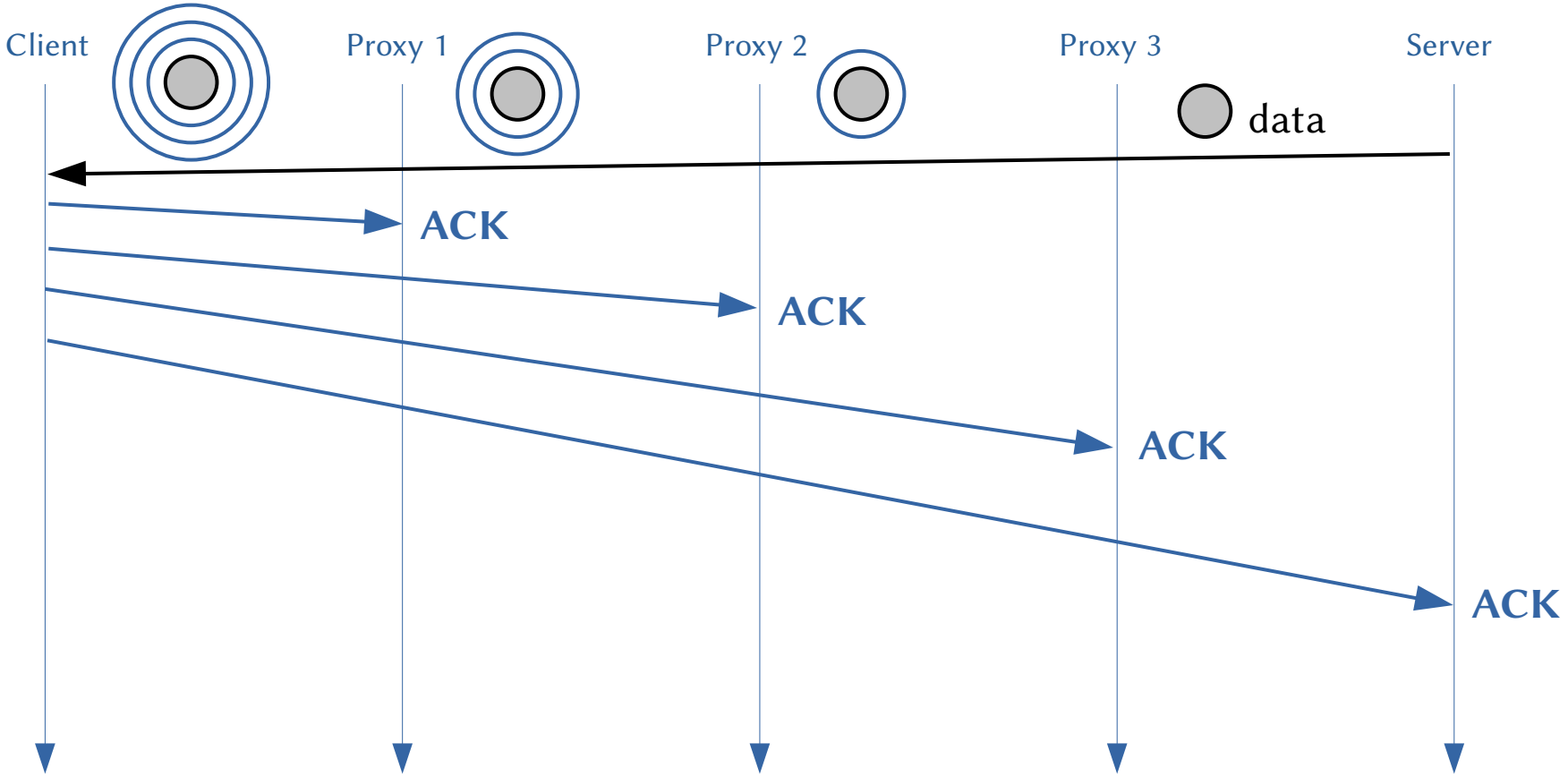
Analysis: Nested ACKs



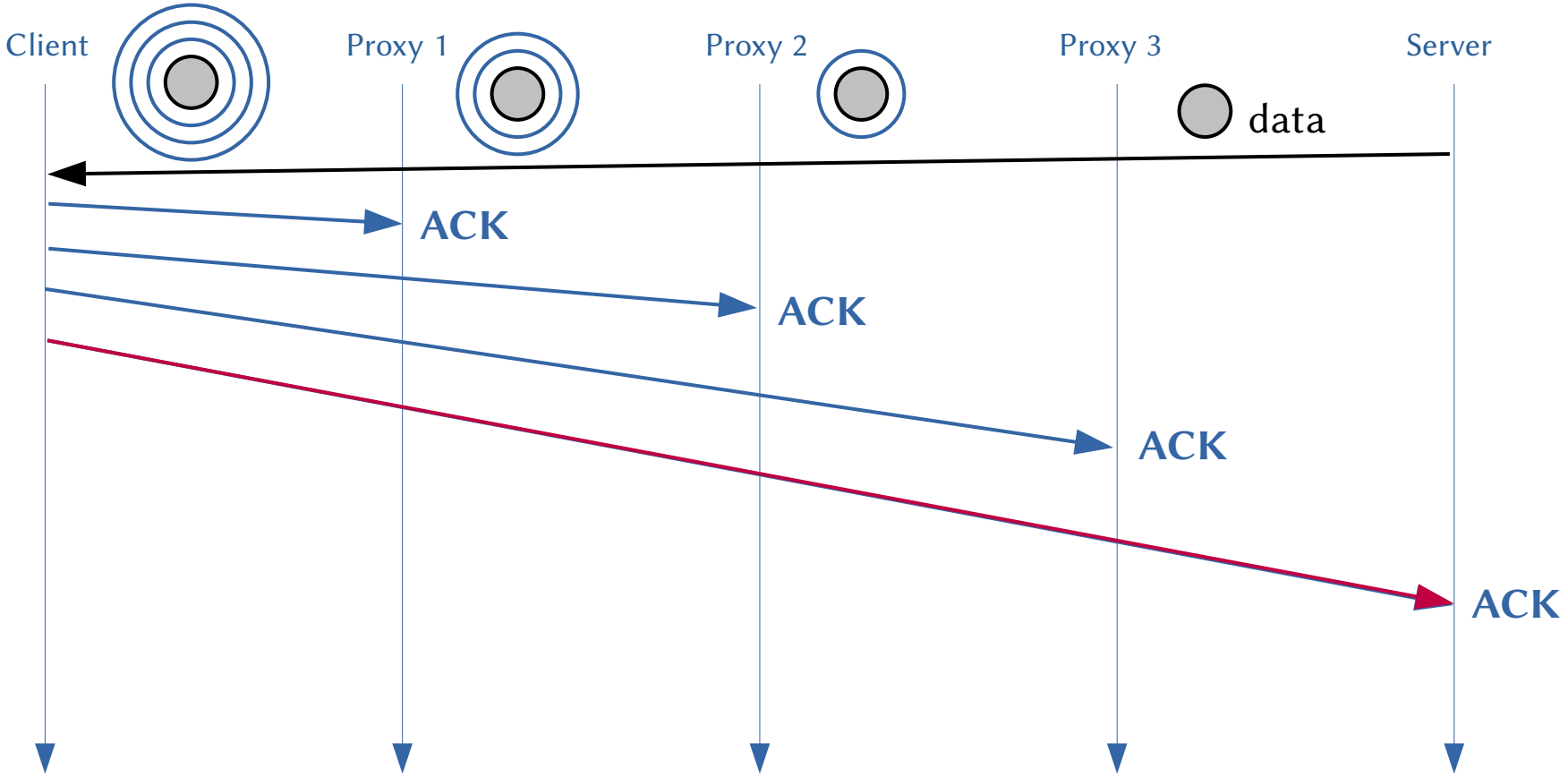
Analysis: Nested ACKs



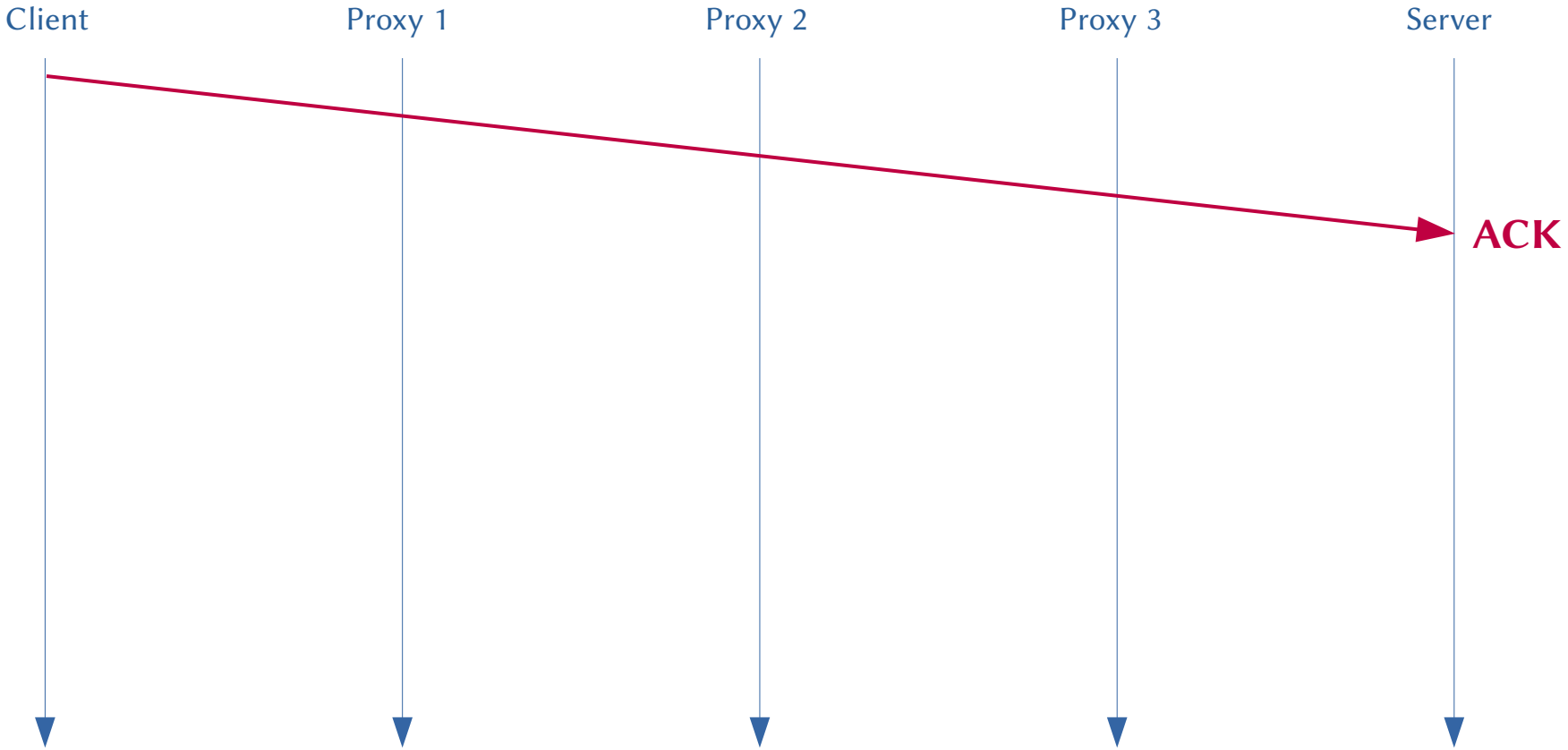
Analysis: Nested ACKs



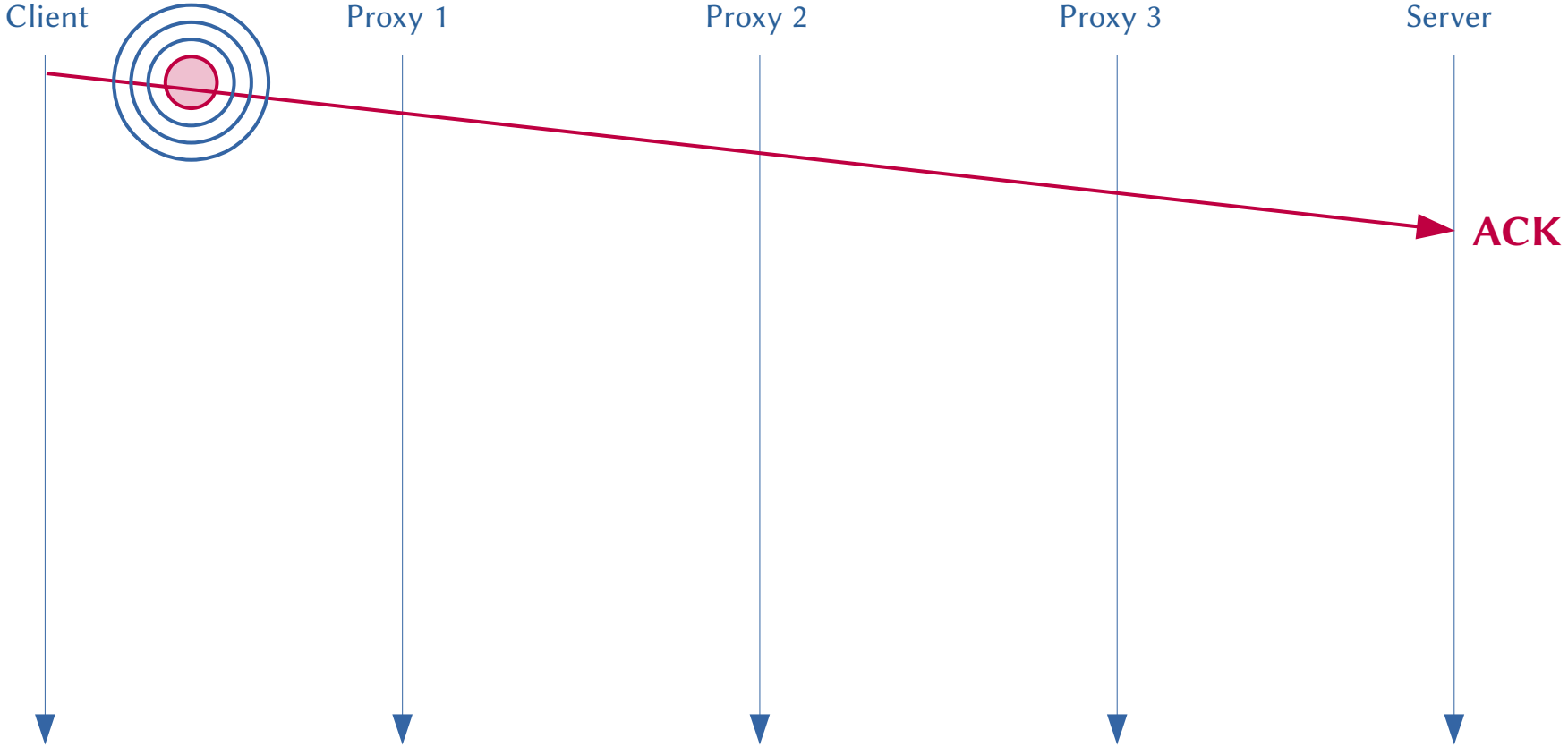
Analysis: Nested ACKs



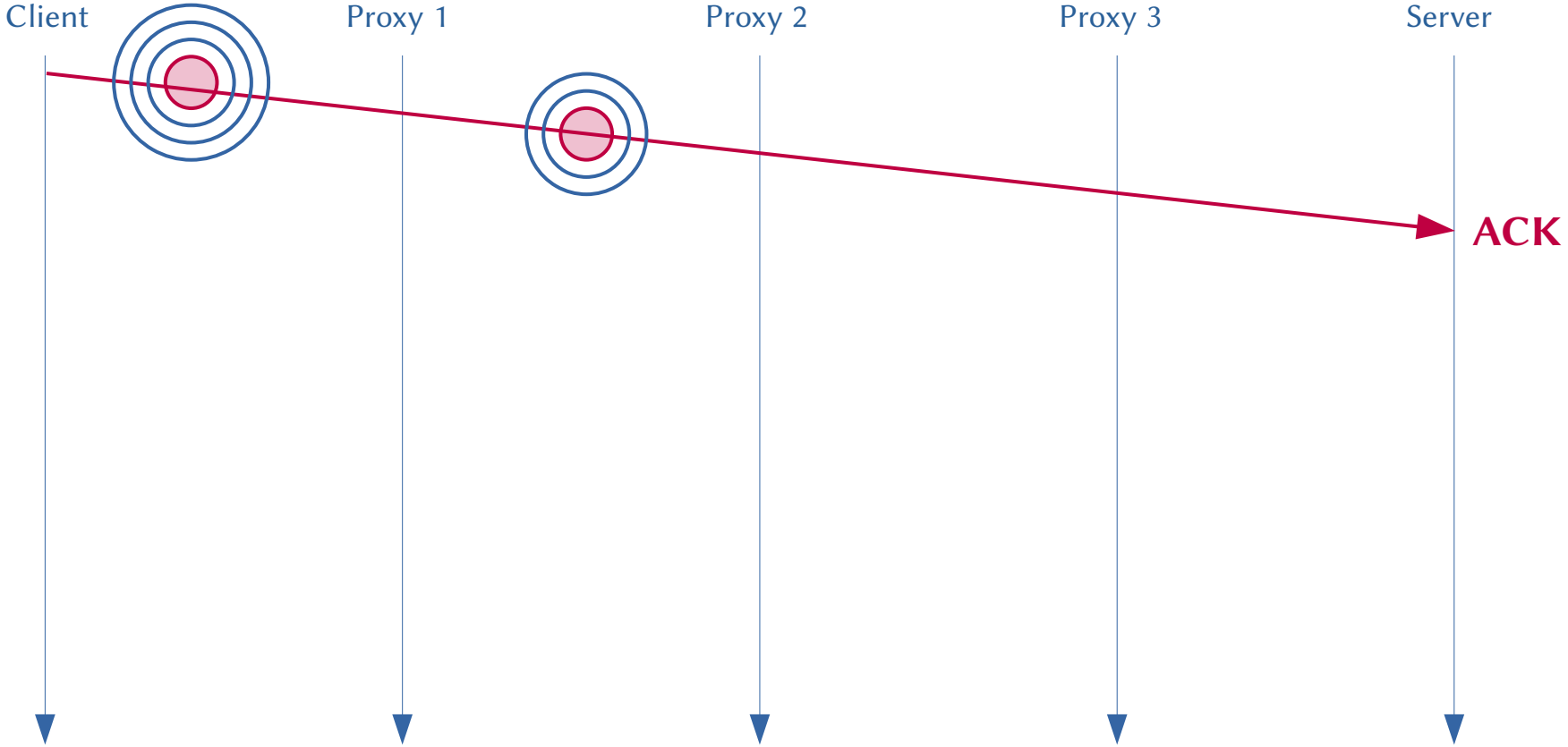
Analysis: Nested ACKs



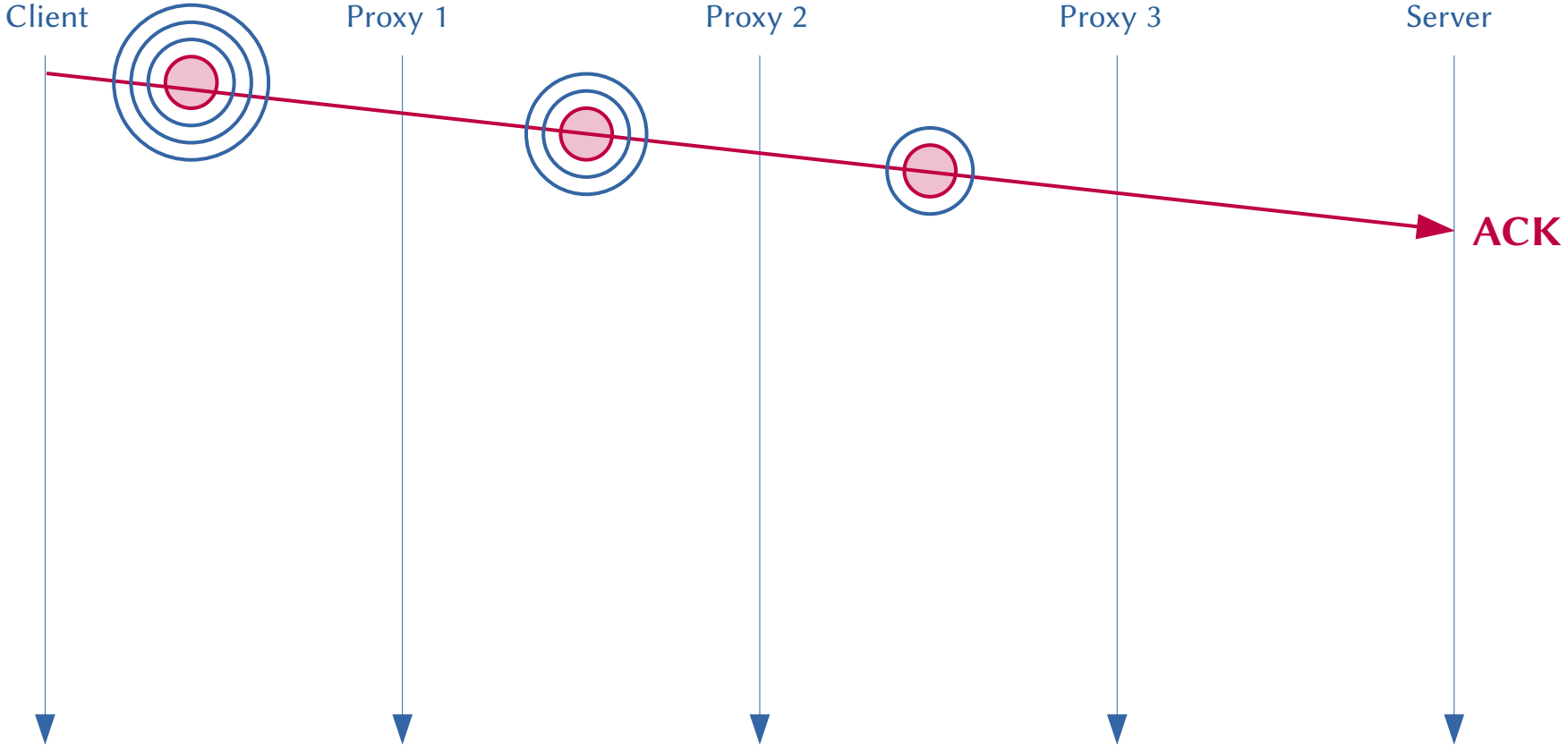
Analysis: Nested ACKs



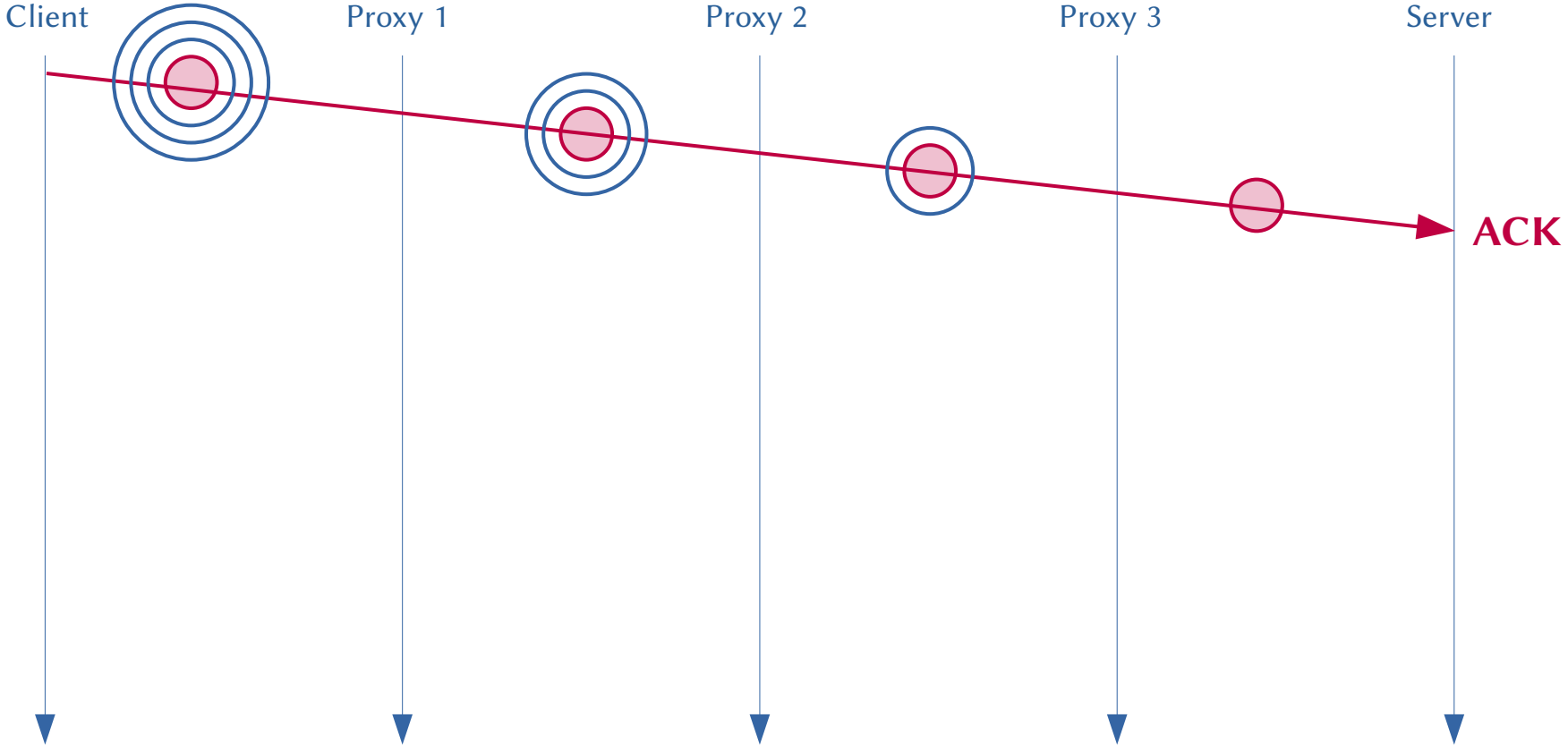
Analysis: Nested ACKs



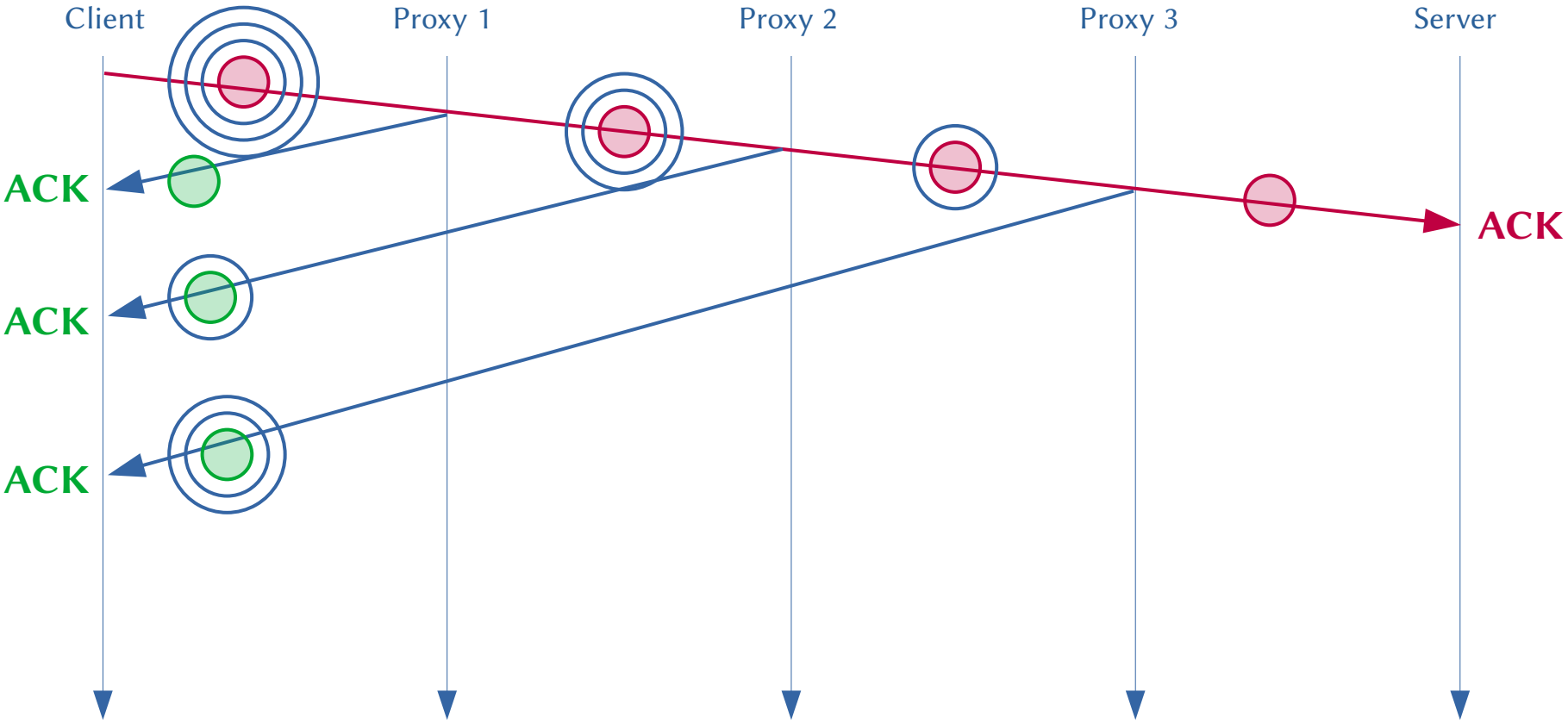
Analysis: Nested ACKs



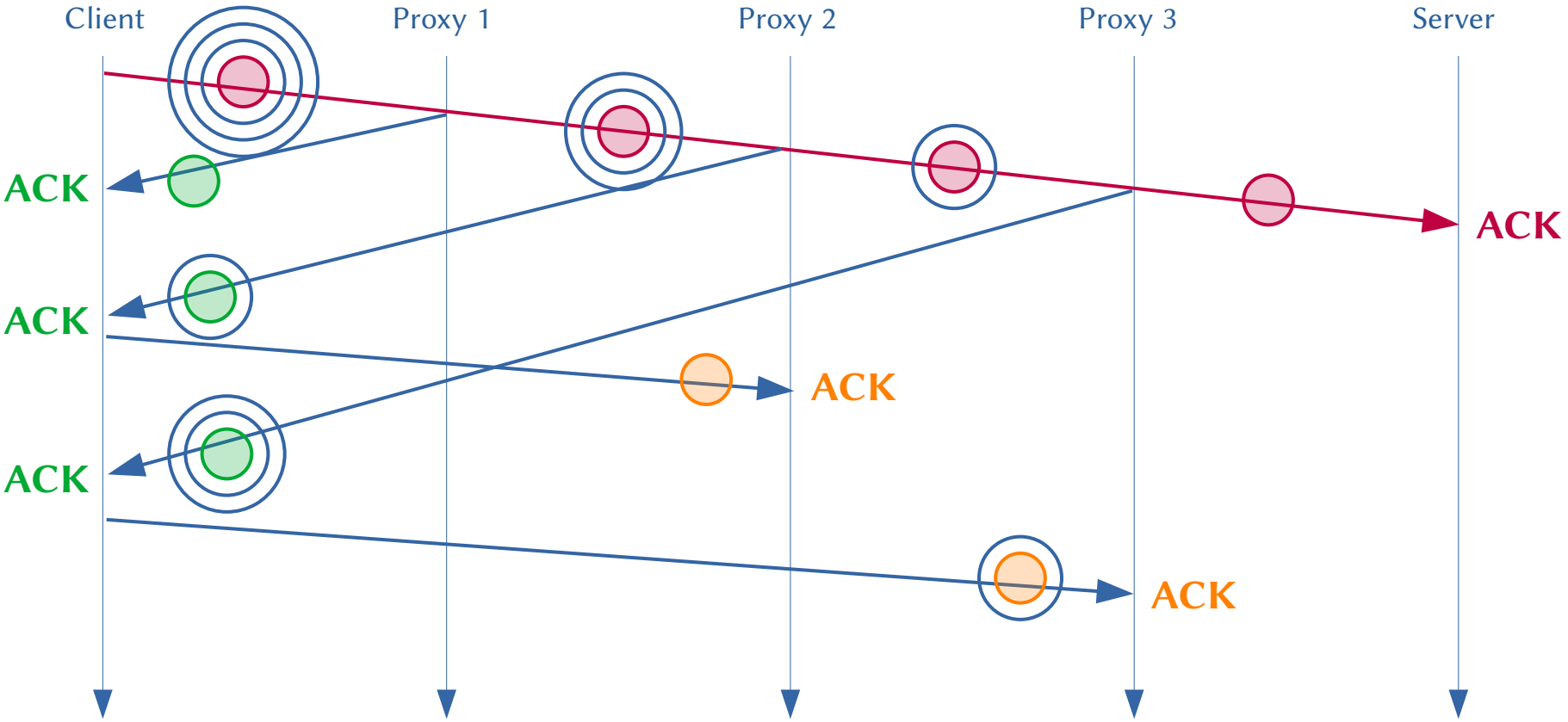
Analysis: Nested ACKs



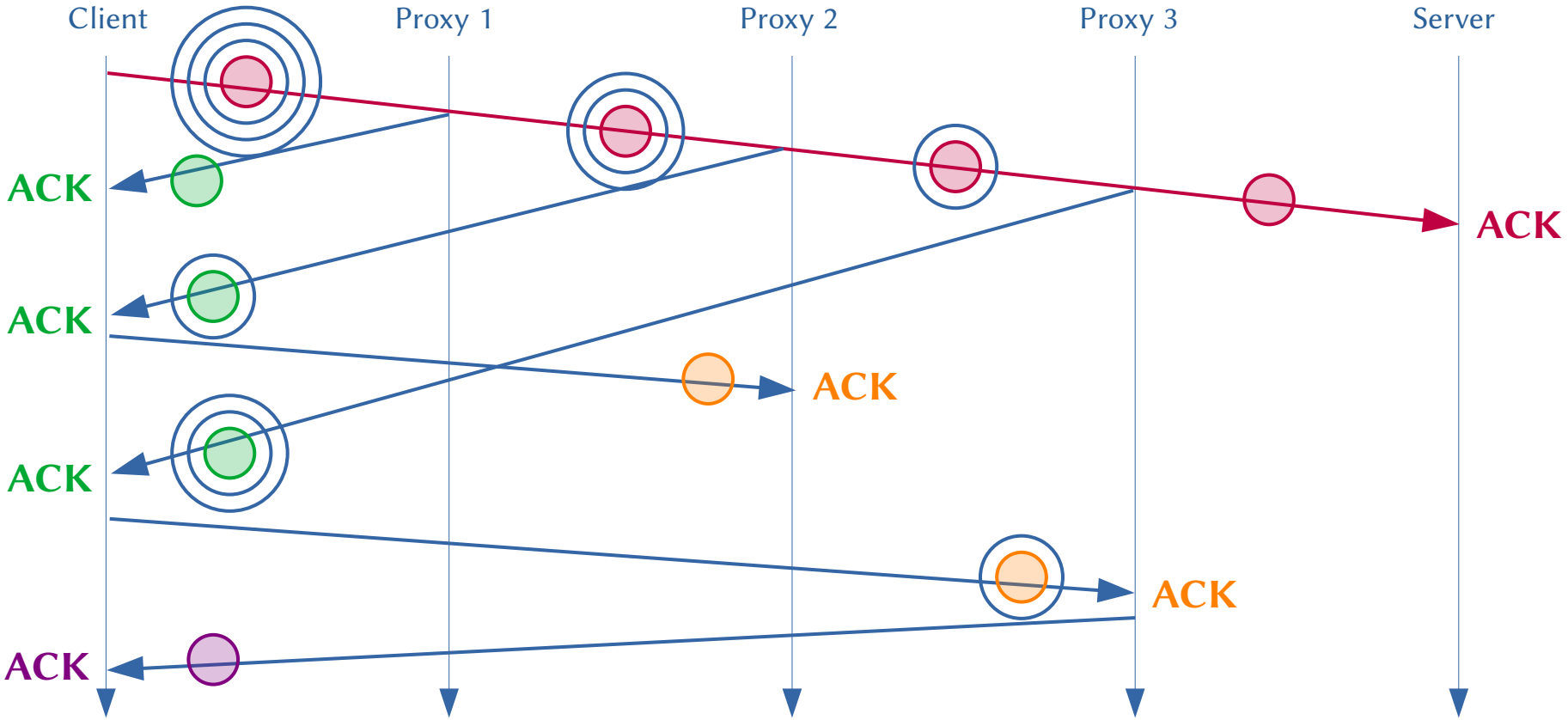
Analysis: Nested ACKs



Analysis: Nested ACKs



Analysis: Nested ACKs



Analysis: Nested ACKs

- Problem Summary: **Encapsulated ACKs become ACK-eliciting themselves.**
  - ➤ Effect escalates with multiple nested tunnels ➤ many ACKs-of-ACKs

Kathrin Elmenhorst

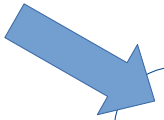
Analysis: Nested ACKs

- Problem Summary: **Encapsulated ACKs become ACK-eliciting themselves.**
  - ➤ Effect escalates with multiple nested tunnels ➤ many ACKs-of-ACKs

- Worst-case impact: Exponential increase (base 2) of ACKs with increasing hops
  - ➤ With **ACK aggregation**: function flattens

Analysis: Nested ACKs

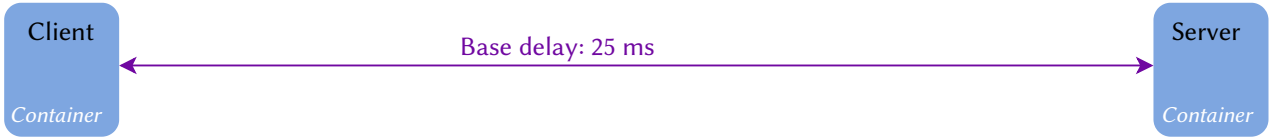
- Problem Summary: **Encapsulated ACKs become ACK-eliciting themselves.**
  - ➤ Effect escalates with multiple nested tunnels ➤ many ACKs-of-ACKs
- Worst-case impact: Exponential increase (base 2) of ACKs with increasing hops
  - ➤ With **ACK aggregation**: function flattens



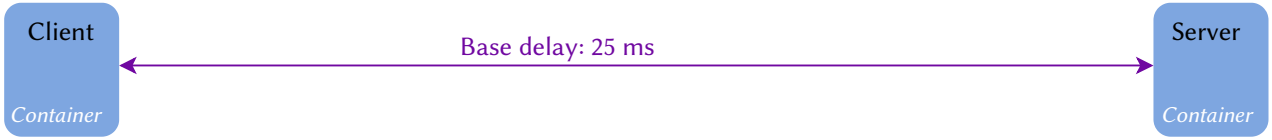
MASQUE ACK aggregation

- Decrease ACK frequency
- Increase packetization efficiency
- Mechanism: **Delay ACKs** until more data to send/ack
  - ➤ Limits: Max. ACK delay, max. ACKed range

Measurement Setup

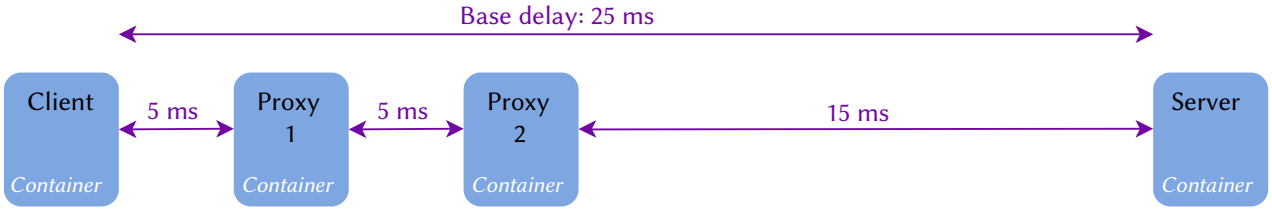


Measurement Setup



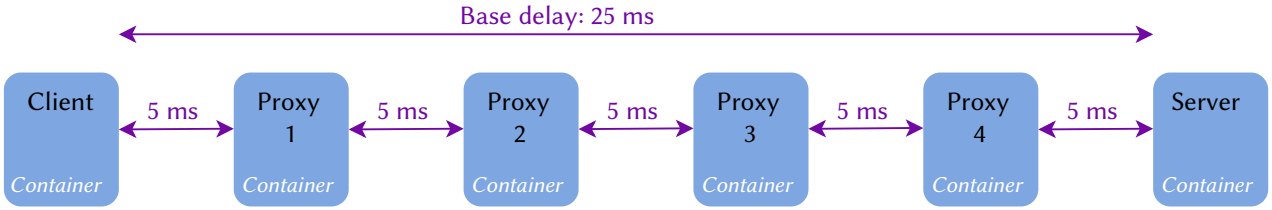
- Separate Docker container for each endpoint
- 0 – 4 proxies

Measurement Setup



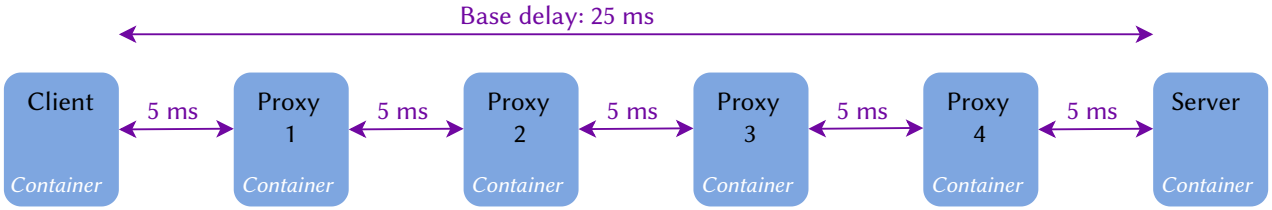
- Separate Docker container for each endpoint
- 0 – 4 proxies

Measurement Setup



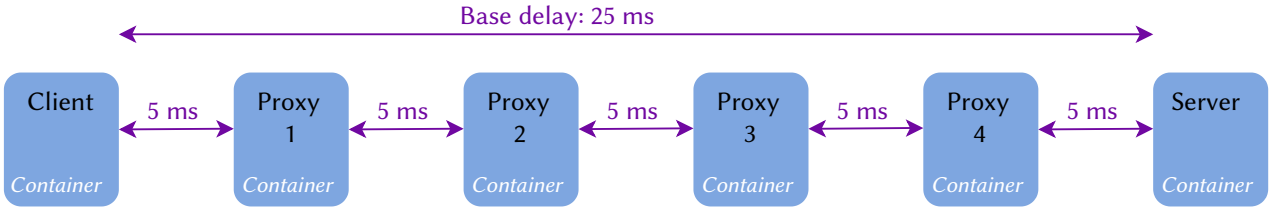
- Separate Docker container for each endpoint
- 0 – 4 proxies

Measurement Setup



- Separate Docker container for each endpoint
- 0 – 4 proxies
- Delay: 25 ms end-to-end
- Bandwidth: 10 / 50 Mbps bottleneck at client

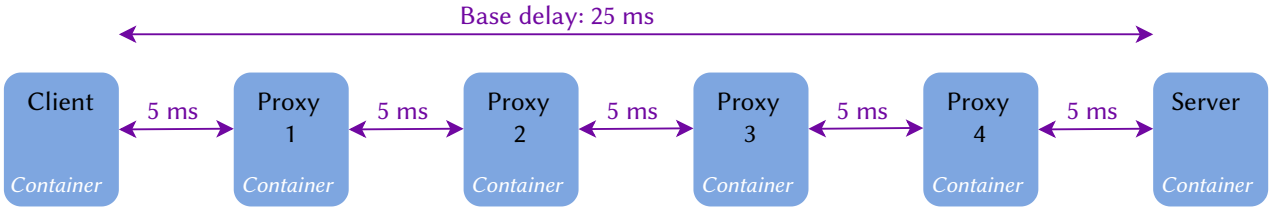
Measurement Setup



- Separate Docker container for each endpoint
- 0 – 4 proxies
- Delay: 25 ms end-to-end
- Bandwidth: 10 / 50 Mbps bottleneck at client

- 2 QUIC stacks: aioquic, Chromium

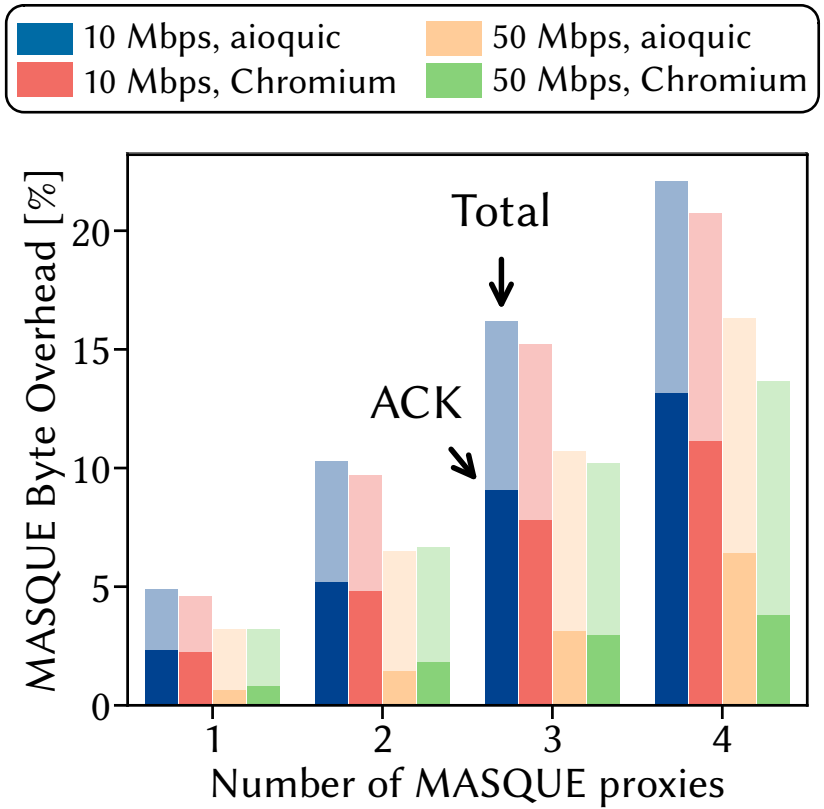
Measurement Setup



- Separate Docker container for each endpoint
- 0 – 4 proxies
- Delay: 25 ms end-to-end
- Bandwidth: 10 / 50 Mbps bottleneck at client

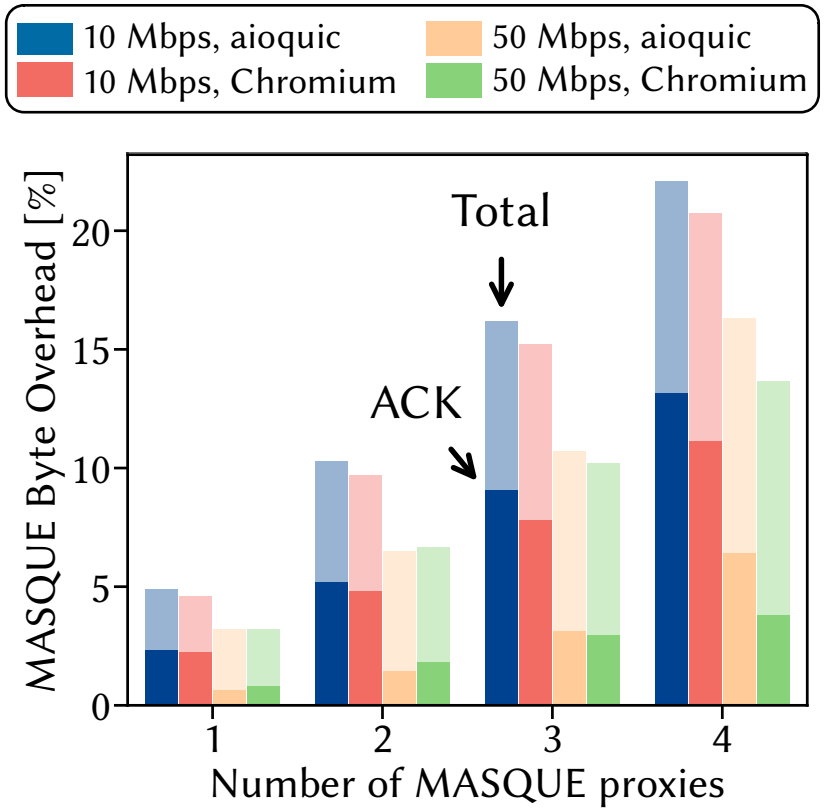
- 2 QUIC stacks: aioquic, Chromium
- Each run: Request 2 MB file via HTTP/3 from target through proxies; 50 repetitions

Measurement Result: Nested ACK Overhead



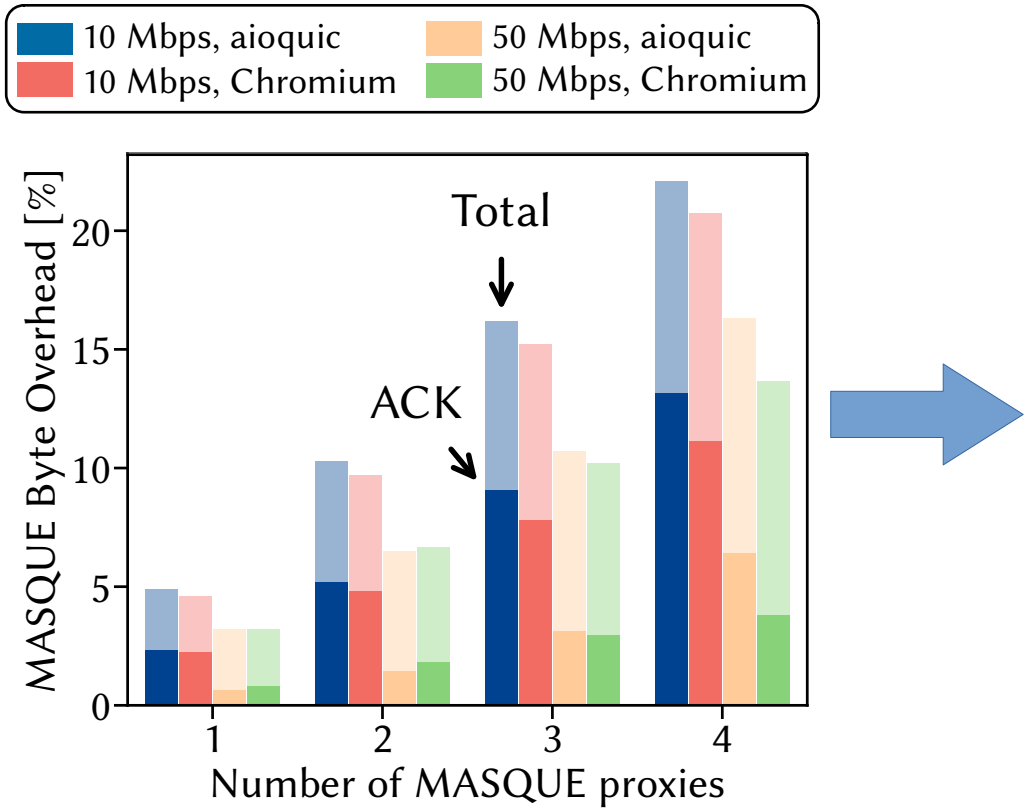
Kathrin Elmenhorst

Measurement Result: Nested ACK Overhead

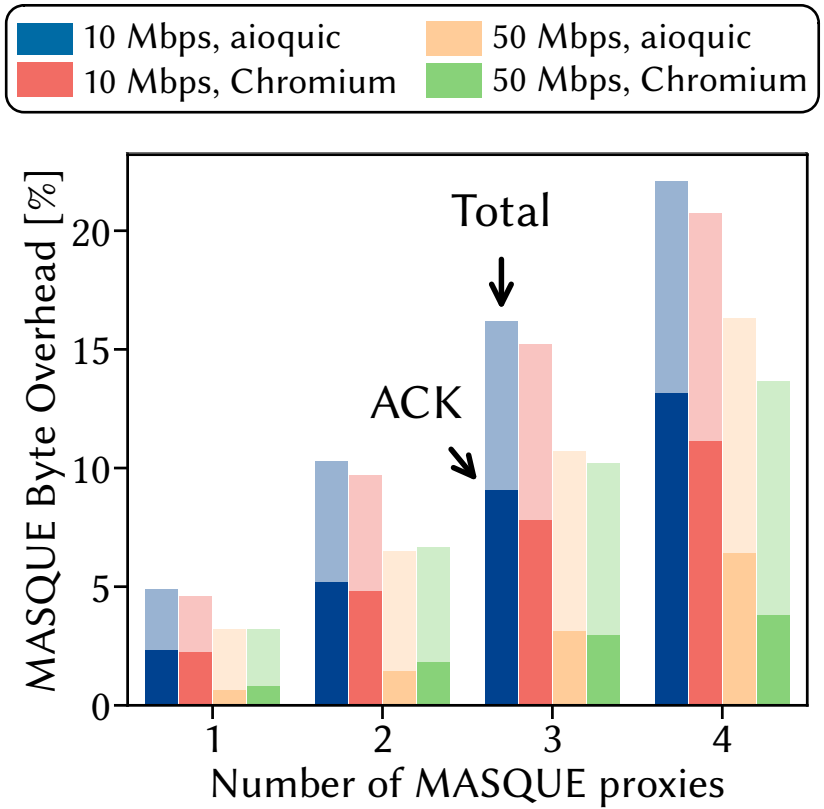


Kathrin Elmenhorst

Measurement Result: Nested ACK Overhead



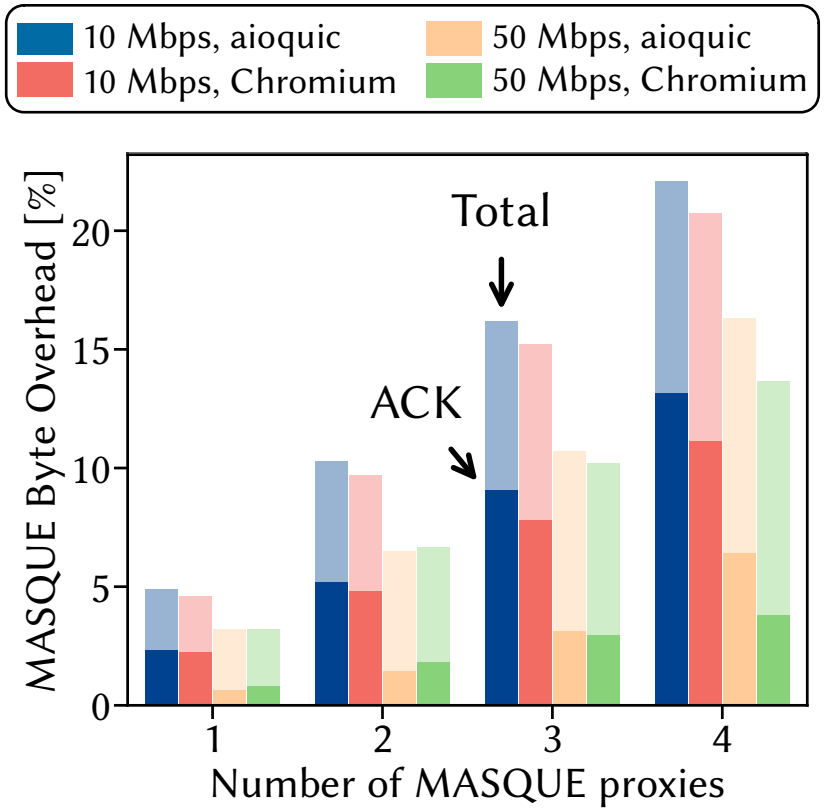
Measurement Result: Nested ACK Overhead



Mitigation approaches:

1. ACK aggregation:
   Delayed ACKs (RFC 9000)
   on tunnel connections

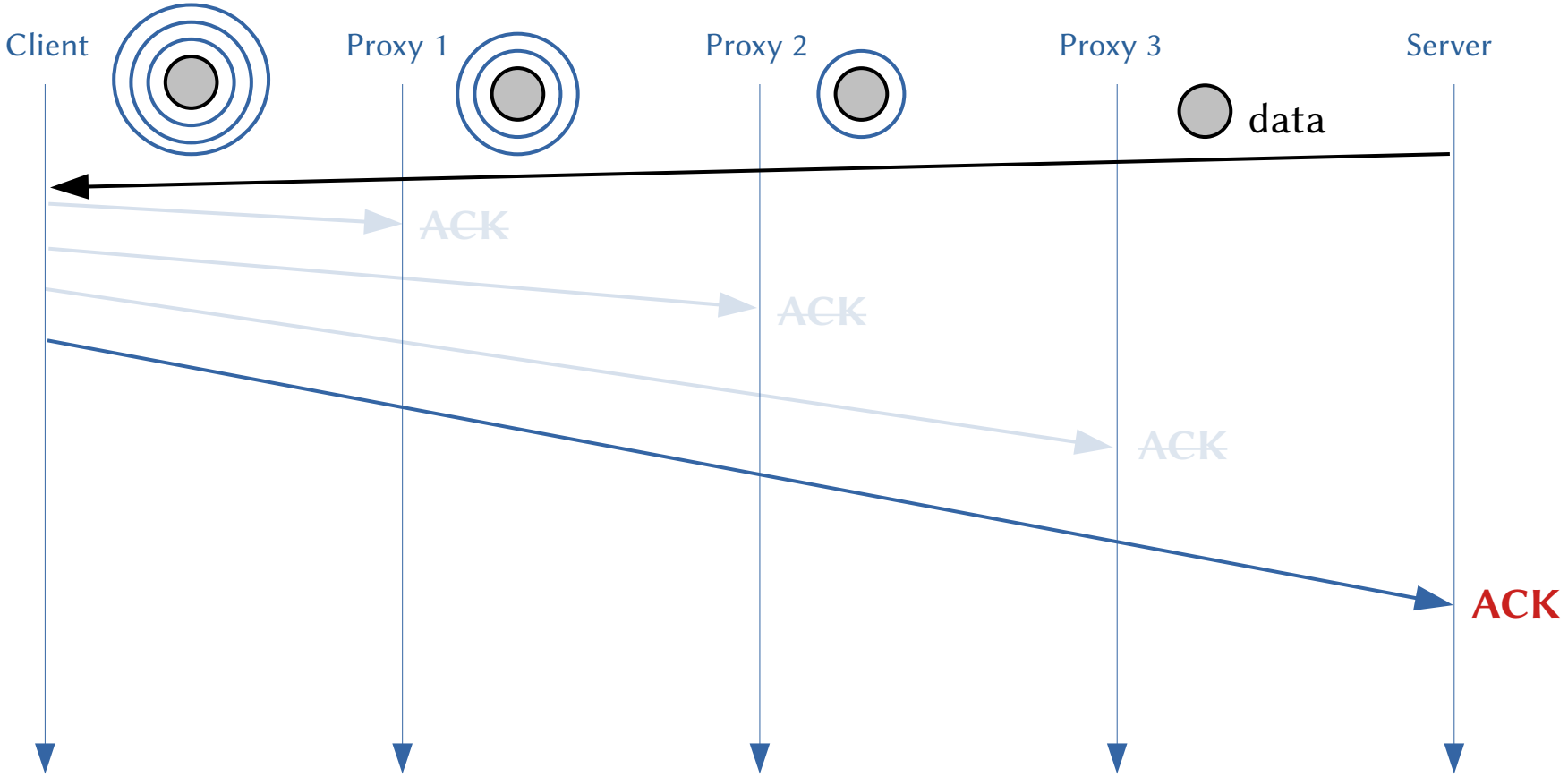
Measurement Result: Nested ACK Overhead



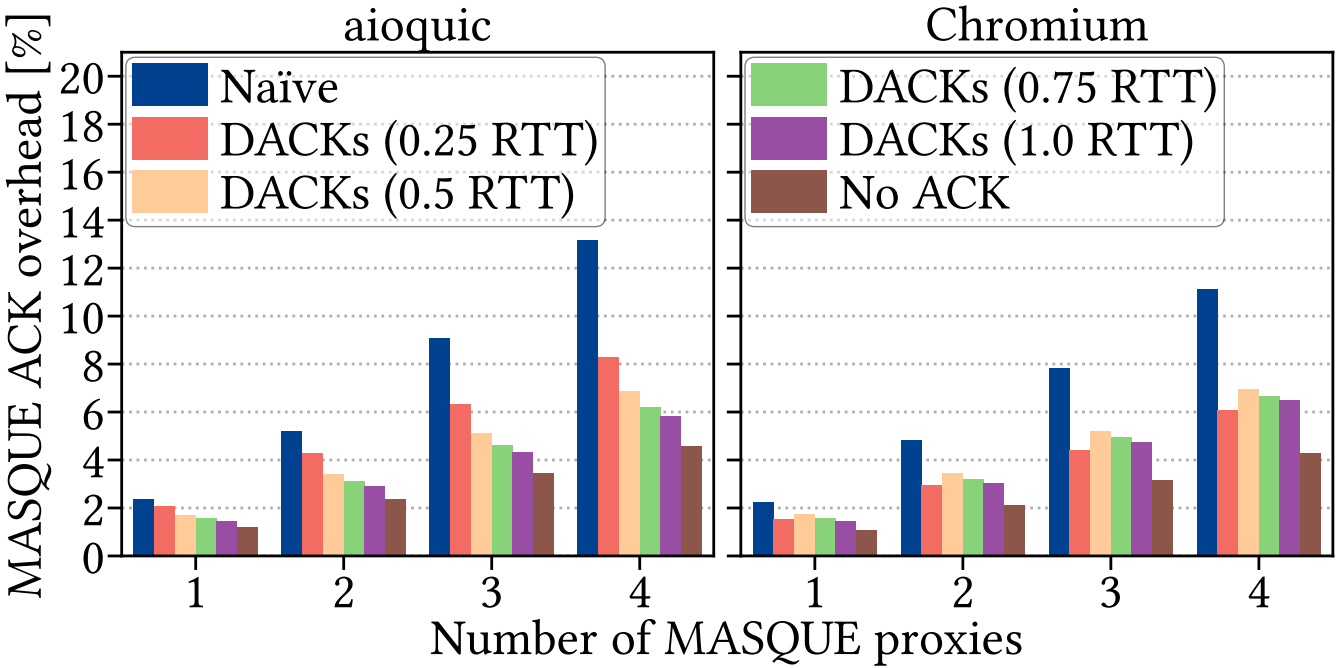
Mitigation approaches:

1. ACK aggregation:
   Delayed ACKs (RFC 9000)
   on tunnel connections
2. ACK elimination:
   Non-ACK-eliciting DATAGRAMs
   - ! Requires disabling of tunnel congestion control

Non-ACK-eliciting DATAGRAM



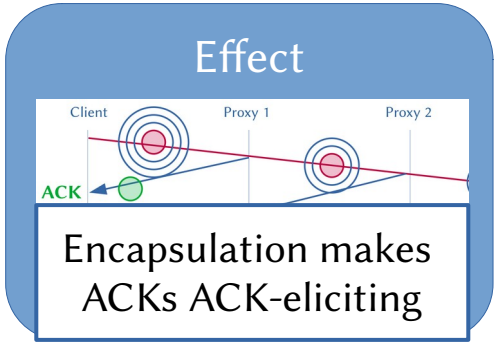
Mitigation Results



Focused on 10 Mbps scenario where the overhead was highest before.

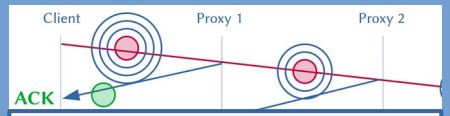
Takeaways and Future Work

Takeaways and Future Work



Takeaways and Future Work

Effect



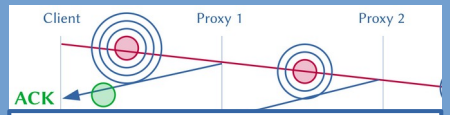
Encapsulation makes ACKs ACK-eliciting

Worst-case impact

Exponential increase of ACKs with the number of proxies

Takeaways and Future Work

Effect

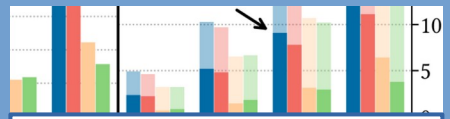


Encapsulation makes ACKs ACK-eliciting

Worst-case impact

Exponential increase of ACKs with the number of proxies

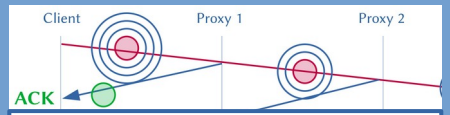
Practical impact



13 % ACK overhead with 4 proxies

Takeaways and Future Work

Effect



Encapsulation makes ACKs ACK-eliciting

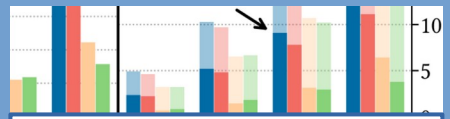
Worst-case impact

Exponential increase of ACKs with the number of proxies

Mitigation potential

++ ACK aggregation on MASQUE tunnel connections

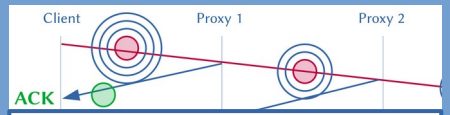
Practical impact



13 % ACK overhead with 4 proxies

Takeaways and Future Work

Effect



Encapsulation makes ACKs ACK-eliciting

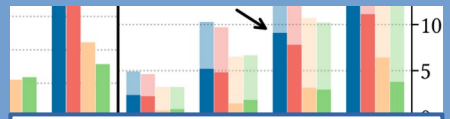
Worst-case impact

Exponential increase of ACKs with the number of proxies

Mitigation potential

++ ACK aggregation on MASQUE tunnel connections

Practical impact



13 % ACK overhead with 4 proxies

Future work

Impact on real-world multi-hop MASQUE deployments

Potential traffic analysis attack against ACK packet size pattern

Appendix

Appendix: References

1. M. Alsabah, et al., “Performance and Security Improvements for Tor: A Survey,” ACM Comput. Surv., vol. 49, 2016.
2. Apple, “iCloud Private Relay Overview,” 2021.
3. **M. Kühlewind, et al., “Evaluation of QUIC-based MASQUE proxying,”** EPIQ, 2021.
4. R. Marx, et. al., "Same Standards, Different Decisions: A Study of QUIC and HTTP/3 Implementation Diversity", EPIQ, 2020.
5. T. Pauly, et al., “QUIC-Aware Proxying Using HTTP,” IETF Internet-Draft, 2024.
6. R. Ramesh, et al., “VPNalyzer: Systematic Investigation of the VPN Ecosystem,” NDSS Symposium, 2022.
7. S. Saleh, et al., “Shedding light on the dark corners of the internet: A survey of Tor research,” Journal of Network and Computer Applications, vol. 114, 2018.
8. **P. Sattler, et al., “Towards a Tectonic Traffic Shift? Investigating Apple’s New Relay Network,”** IMC, 2022.
9. D. Schinazi, et al., “HTTP Datagrams and the Capsule Protocol,” RFC 9297, 2022.
10. D. Schinazi, “Proxying UDP in HTTP,” RFC 9298, 2022.
11. **M. Trevisan, et al., “Measuring the Performance of iCloud Private Relay,”** Passive and Active Measurement Conference, 2023.
12. **A. Zohaib, et al., “Investigating Traffic Analysis Attacks on Apple iCloud Private Relay,”** ACM Asia CCS, 2023.

Appendix: Formalization of worst-case overhead

Recursive definition:

$$a_r(0) = \frac{1}{r}$$

$$a_r(n) = a_r(n-1) + \frac{1}{r} \cdot (1 + a_r(n-1))$$

Closed-form definition:

$$\hat{f}_r(n) = \frac{1}{r} \cdot \sum_{i=0}^{n} \left(1 + \frac{1}{r}\right)^i$$

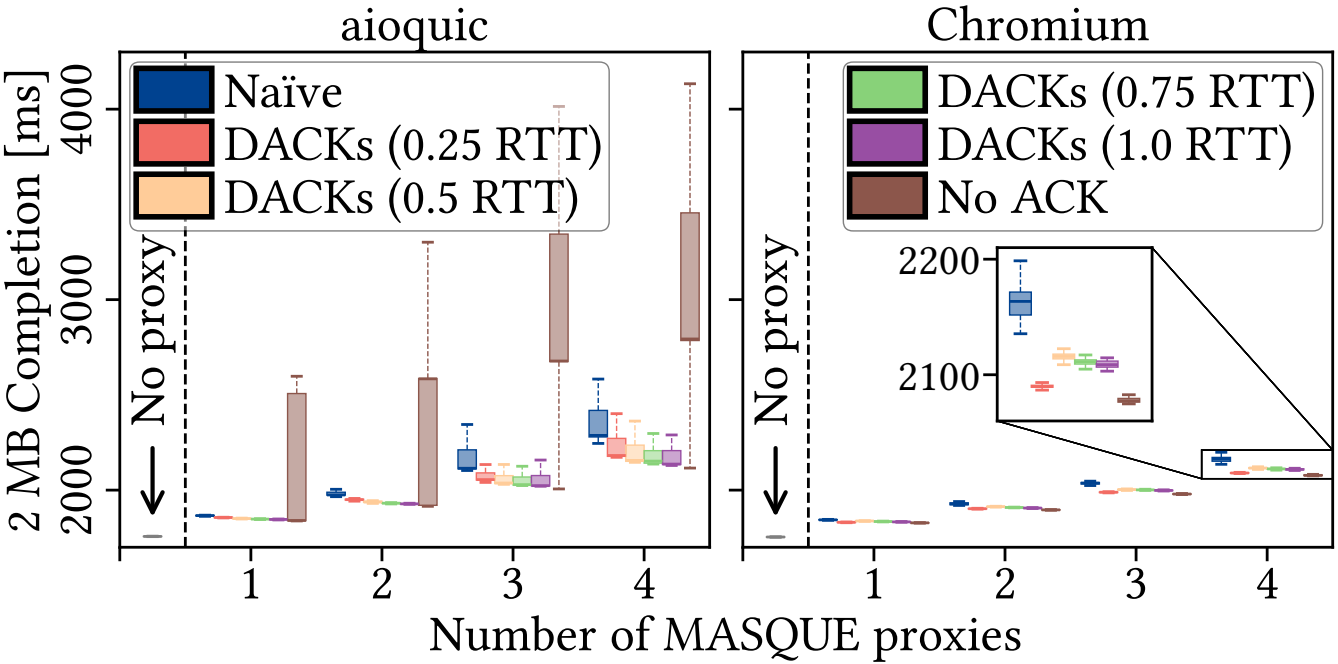
Without innermost end-to-end ACKs:

$$f_r(n) = \frac{1}{r} \cdot \sum_{i=1}^{n} \left(1 + \frac{1}{r}\right)^i$$

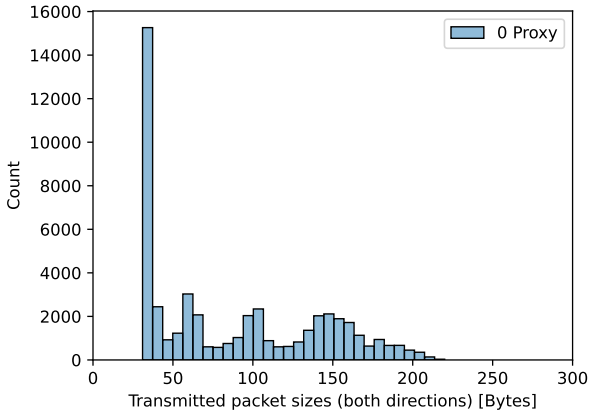
Without ACK aggregation, i.e., r = 1:

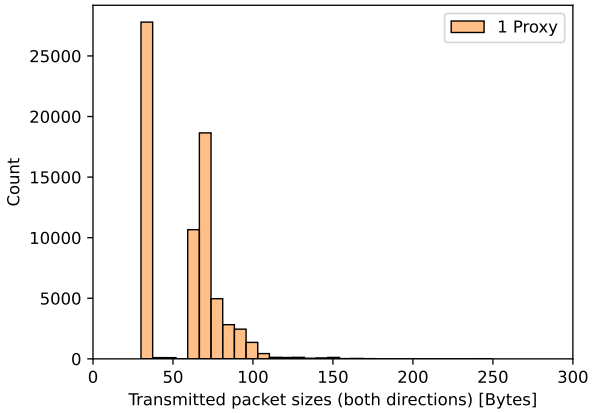
$$f_1(n) = \sum_{i=1}^{n} 2^i$$

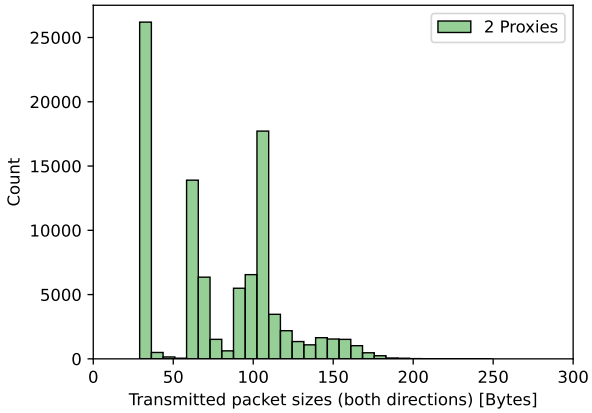
Appendix: Impact of ACK delay on download completion time

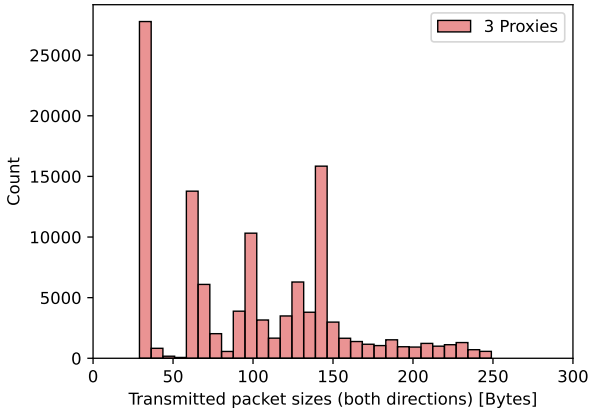


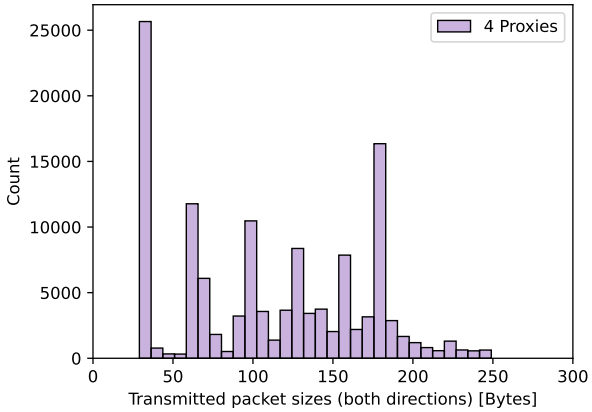
Appendix: Privacy Implications











Packet size frequencies for smaller packets measured at first link (all tunnels).

10 Mbps scenario, Chromium