Encryption in the transport layer: QUIC, TLS1.3, and HTTP/2

Mirja Kühlewind <mirja.kuehlewind@tik.ee.ethz.ch>

Brian Trammell <trammell@tik.ee.ethz.ch>

SwiNOG

May 24, 2018

mami
measurement and architecture for a middleboxed internet

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688421.The opinions expressed and arguments employed reflect only the authors' view. The European Commission is not responsible for any use that may be made of that information.

Supported by the Swiss State Secretariat for Education, Research and Innovation under contract number 15.0268. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the Swiss Government.

Overview
Encryption in the web

- QUIC - New, encrypted transport protocol
  - Optimized for HTTP/2 as an alternative to TCP
  - Designed to be implemented in user space, e.g. in your browser

- TLS 1.3 and DTLS - improved end-to-end encryption
  - Perfect Forward Secrecy (Static RSA and Static Diffie-Hellman cipher suites removed)

- HTTPs uses TLS/TCP or QUIC

Why encryption?

1. Improvements in data and privacy protection (Snowden revelations)
   - "Encrypt it all!“
2. Reaction to ossification
   - Endpoint control about exposure to maintain evolution
   - Increase transparency on in-network handling
   - Middlebox traversal
   - Implementation correctness

Overview

HTTP/2

- Published as IETF RFC7540 in May 2015
- HTTP/2 is based on SPDY as originally proposed by Google
- About 250K domains support HTTP/2 (e.g. see isthewebhttp2yet.com)
  - Deployment driven by big hosting providers

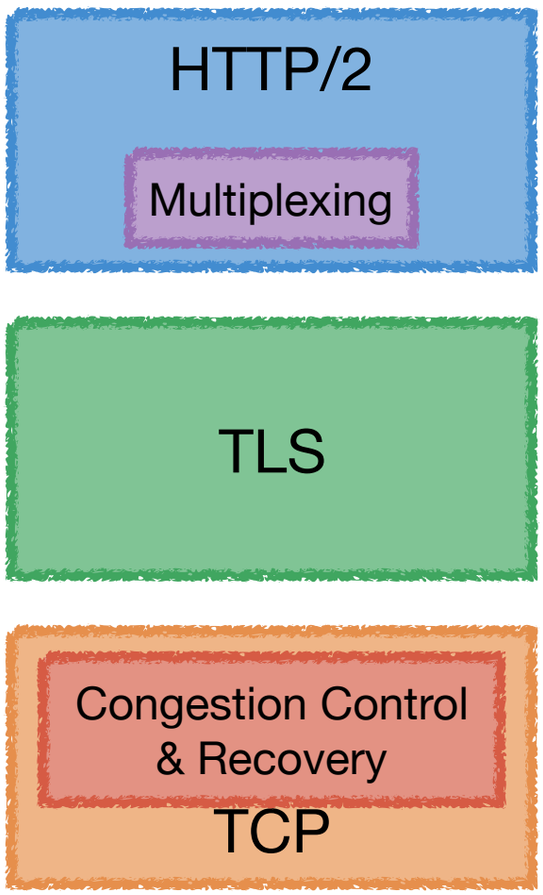
QUIC

- Currently under standardization in the IETF (since Nov'16)
- Originally developed by Google together with SPDY

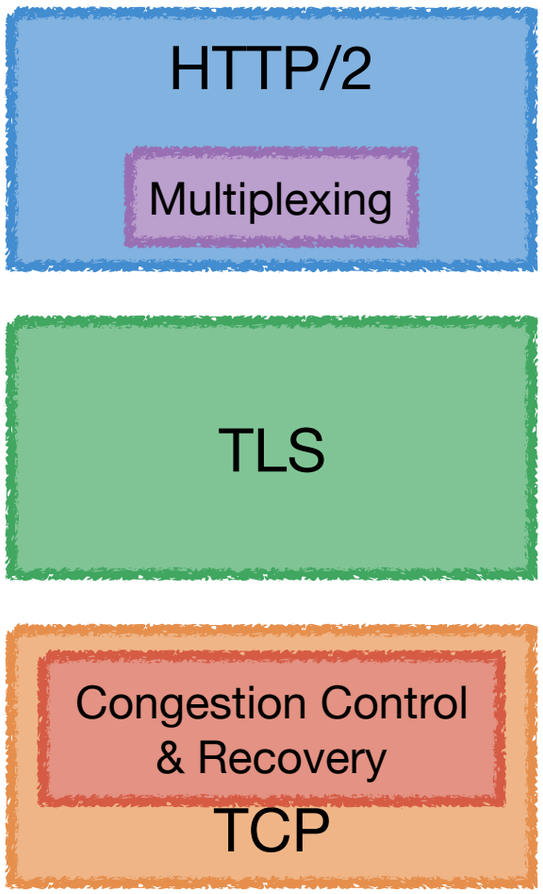
TLS1.3

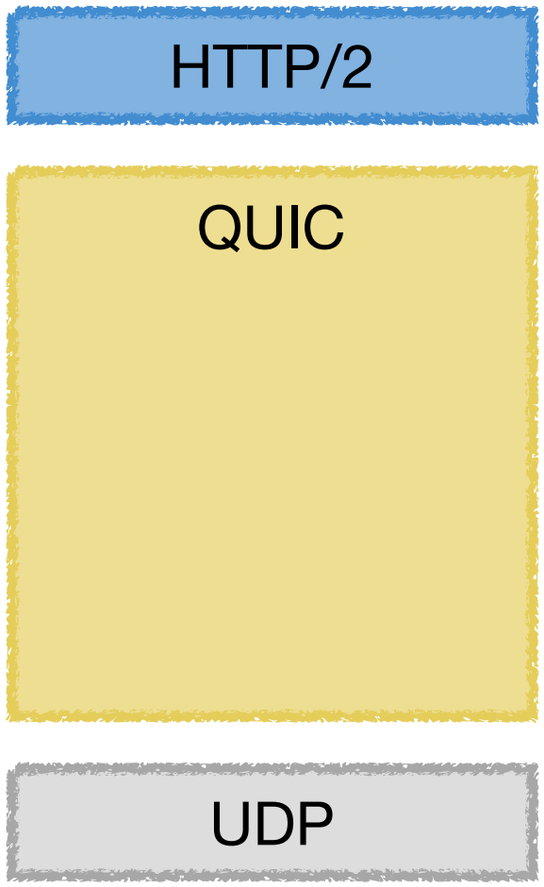
- draft-ietf-tls-tls13 (approved for publication by IESG)
- used by QUICv1 as build in encryption protocol

HTTP/2 over QUIC

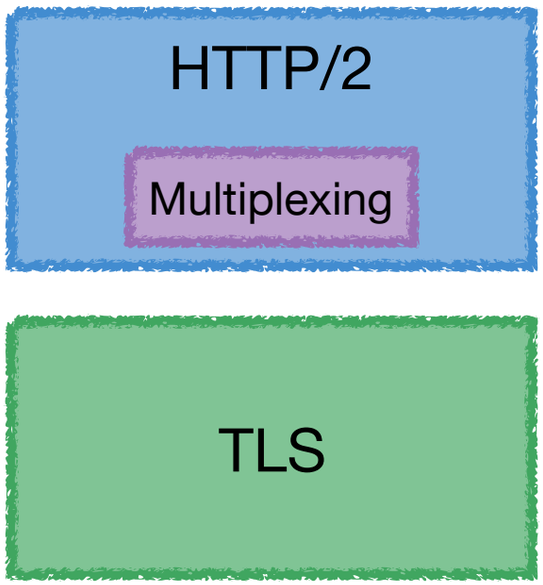


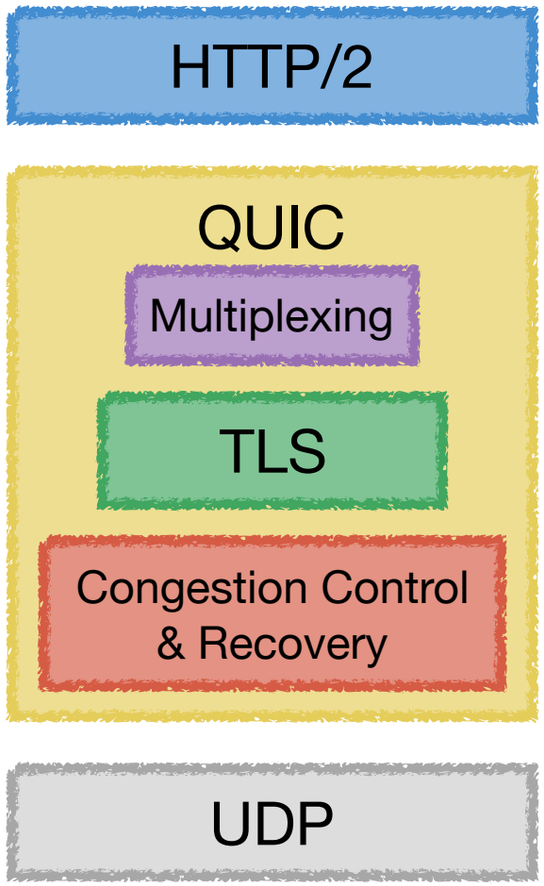
HTTP/2 over QUIC





HTTP/2 over QUIC





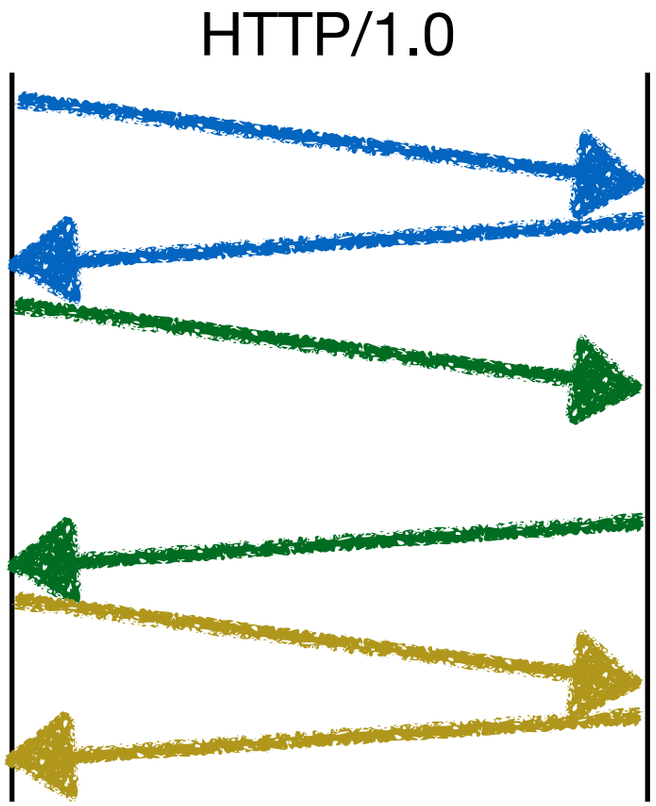
- Stream multiplexing avoids HoL of independent data resources
- Always authenticated and payload is fully encrypted

➡ **Stream & other control information (for e.g. recovery) is not visible to the network anymore!**

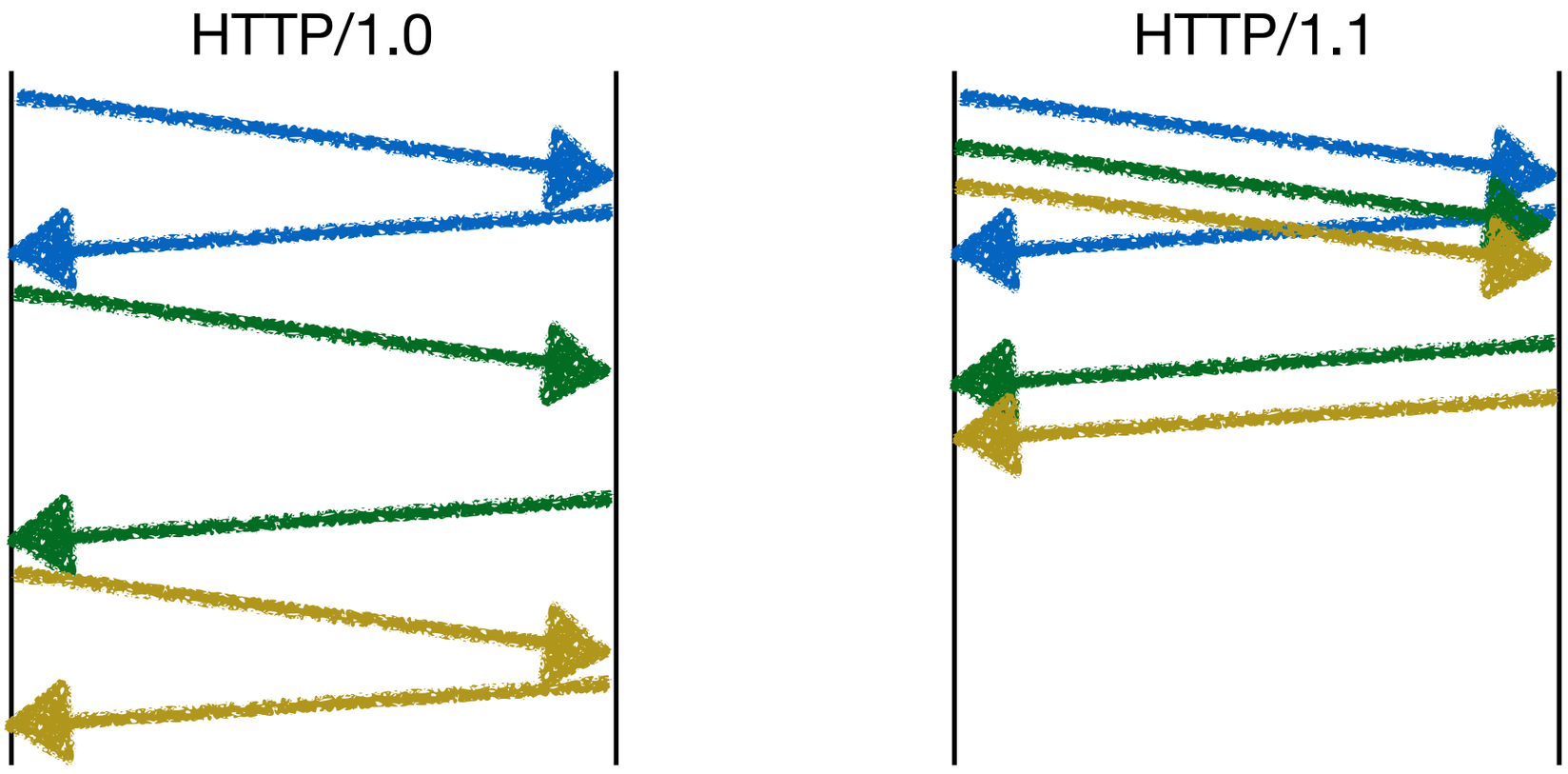
HTTP/2 - Features

- New features to reduce latency of page load time
  - Header compression (HPACK)
  - Server Push
  - Multiplexing of multiple requests over 1 TCP connection
    - avoids Head of Line (HoL) blocking

HTTP/1.1 — Pipelining

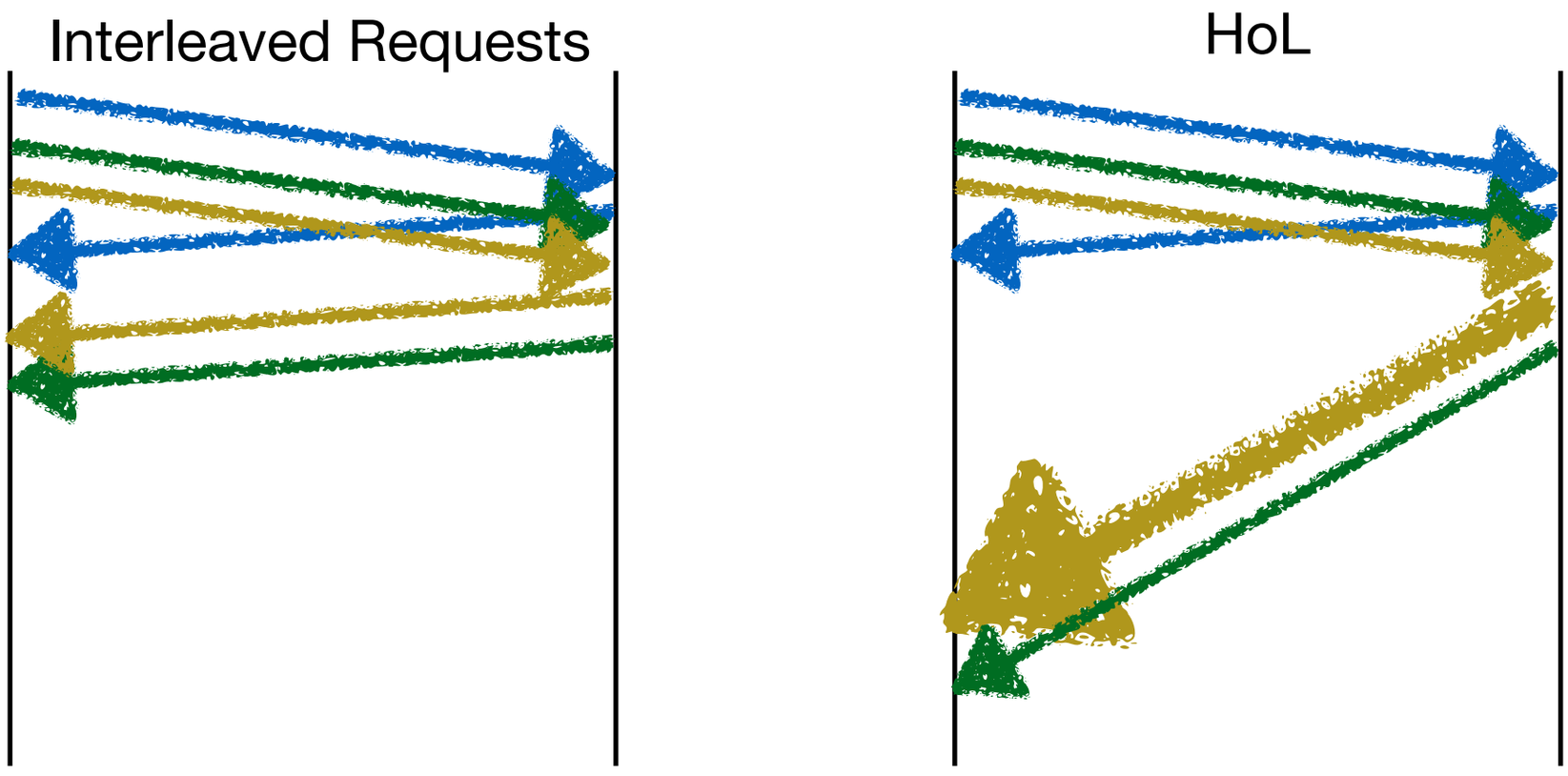


HTTP/1.1 — Pipelining



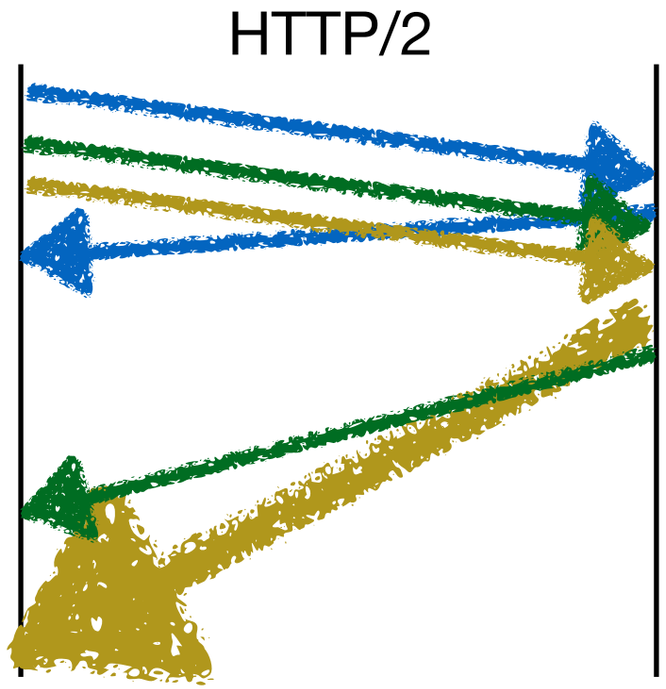
- With pipelining multiple requests can be sent at once
  - ➡ All requests and responses can be sent over the same TCP connection

HTTP — Head of Line Blocking (HoL)



- Large resources can delay later, more important resources
  - ➡ Head-of-Line (HoL) blocking is still possible

HTTP/2 — Stream Multiplexing



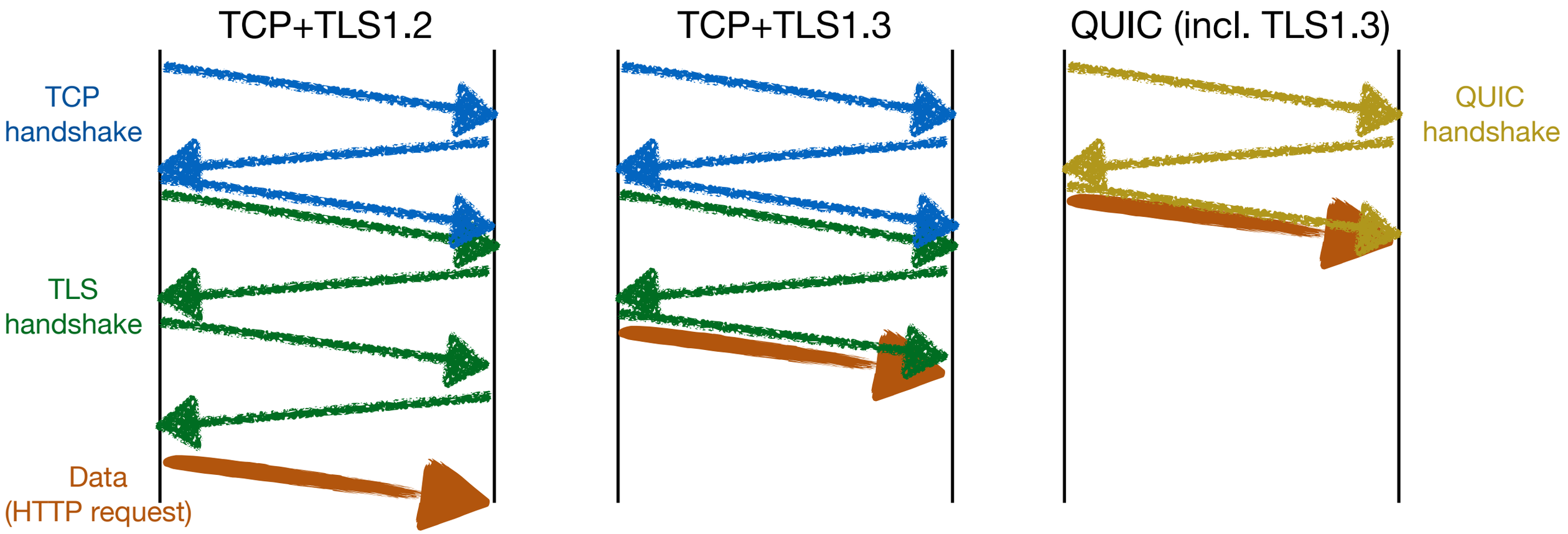
- Requests are send on independent streams in frames

  ➡ Loss or reordering can lead to Head-of-Line (HoL) blocking in the transport layer/TCP

QUIC - Features

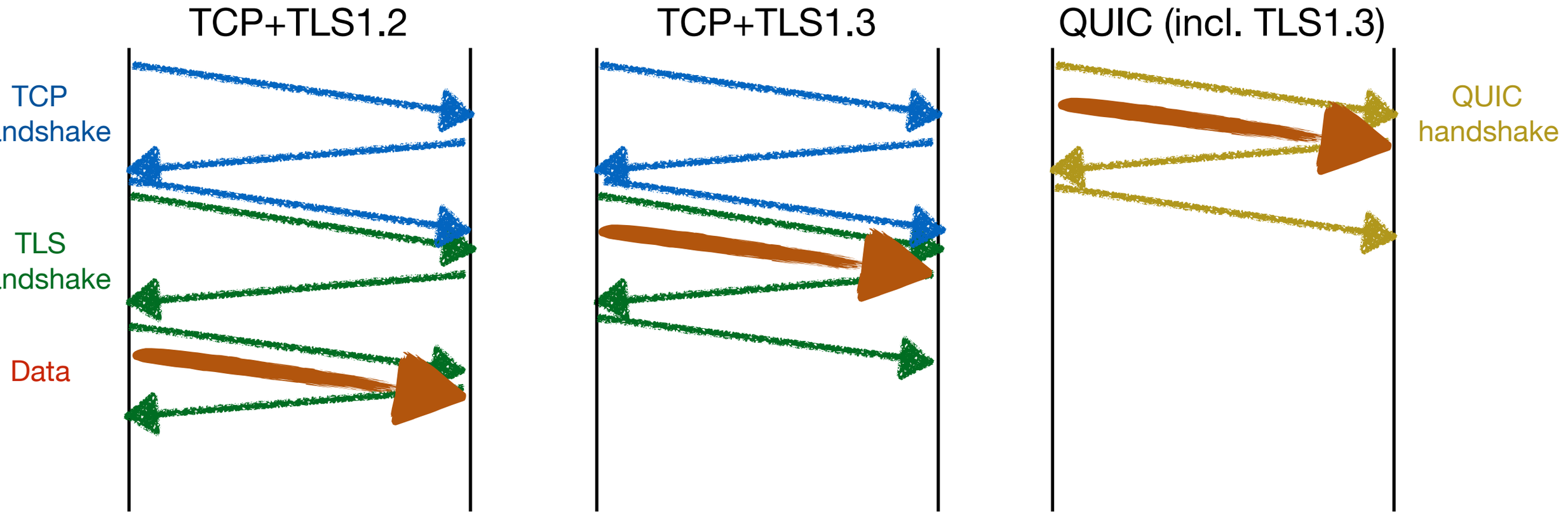
- Stream multiplexing
- Ricer feedback information for enhanced congestion control
- Inherently encrypted using TLS1.3 handshake and control
- 0-RTT low latency support

QUIC — Low Latency Session Establishment



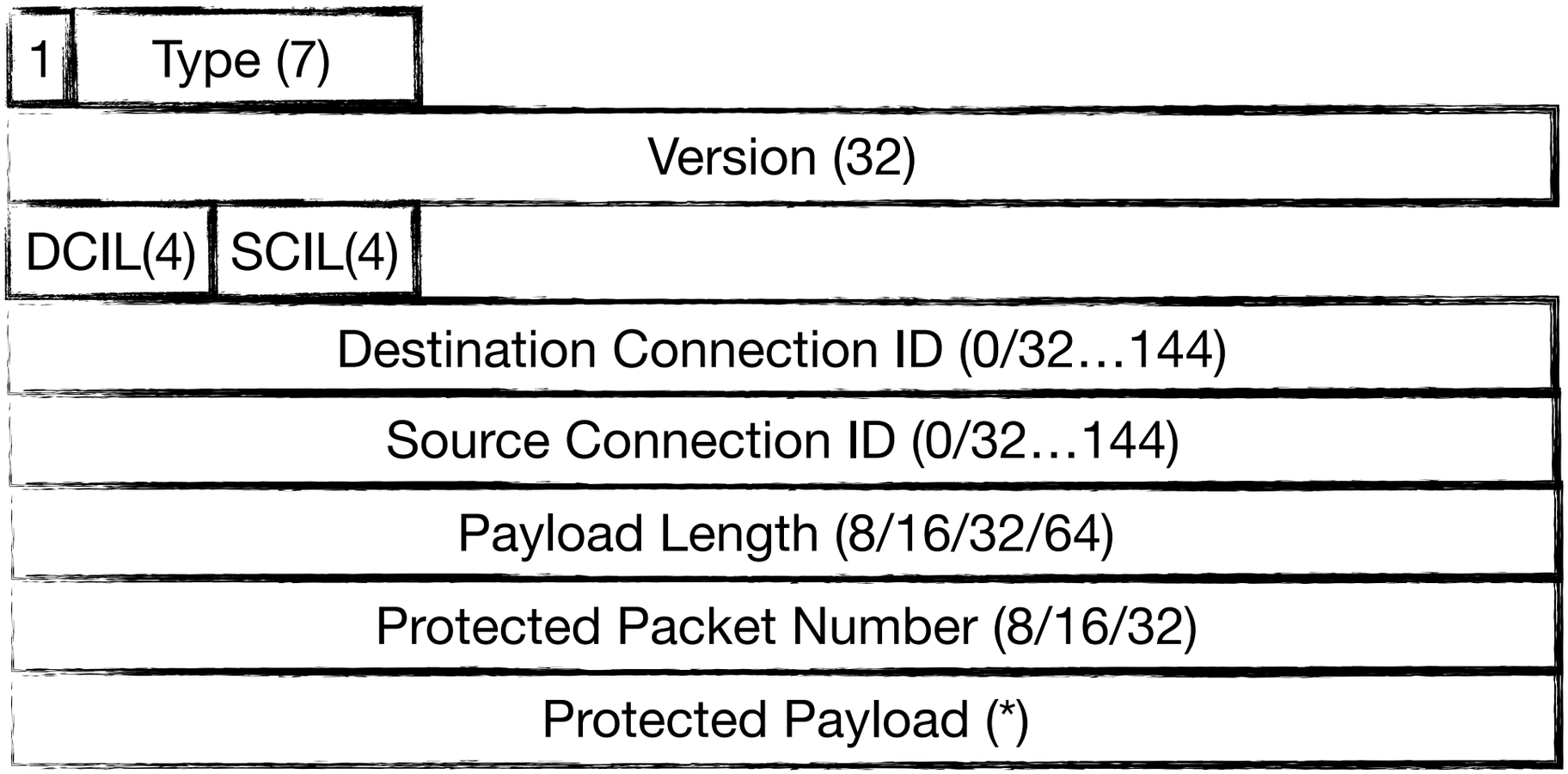
- QUIC combines the transport and crypto handshake
- Handshake packets are not encrypted but verified later

QUIC — 0-RTT Session Resumption



- QUIC can send encrypted payload data in the first RTT on session resumption to a previously connected server
- Care must be taken as QUIC/TLS1.3 0-RTT data are not protected against reply attacks

The QUIC wire image — Long header format (handshake & 0-RTT)



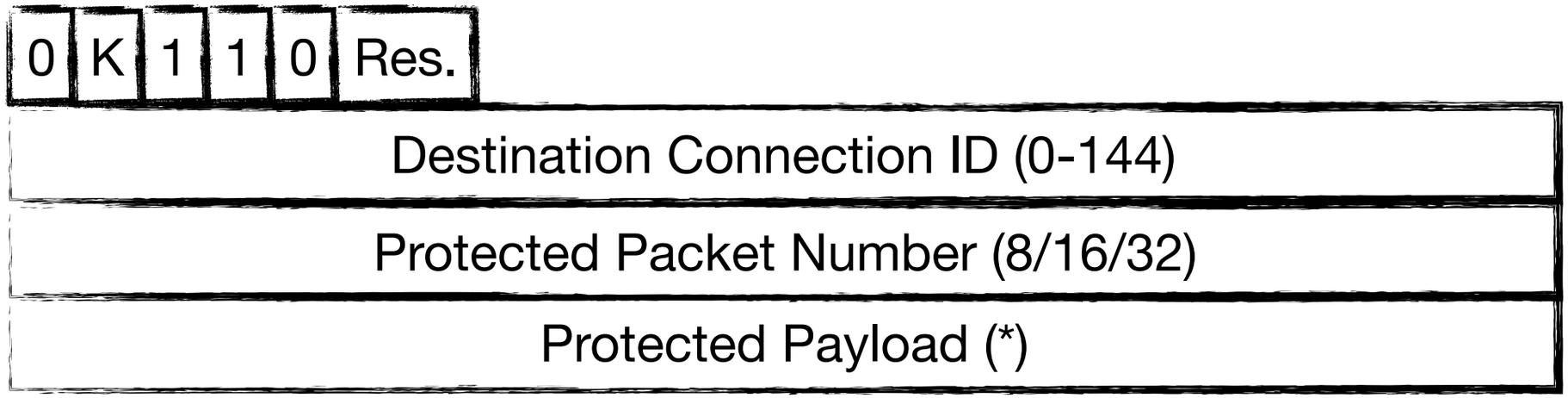
- (Optional) Connection ID for migration and resilience to NAT rebinding
- Packet number is confidentiality protected separate from packet protection (as used as cryptographic nonce for packet encryption)

QUIC Long Header Types

- **Cleartext** packets for (Client) Initial, (Server Stateless) Retry, and (Server/Client) Handshake
- **Encrypted** payload for 0-RTT

Type	Name
0x7F	Initial
0x7E	Retry
0x7D	Handshake
0x7C	0-RTT **Protected**

The QUIC wire image — Short header format (only 1-RTT)



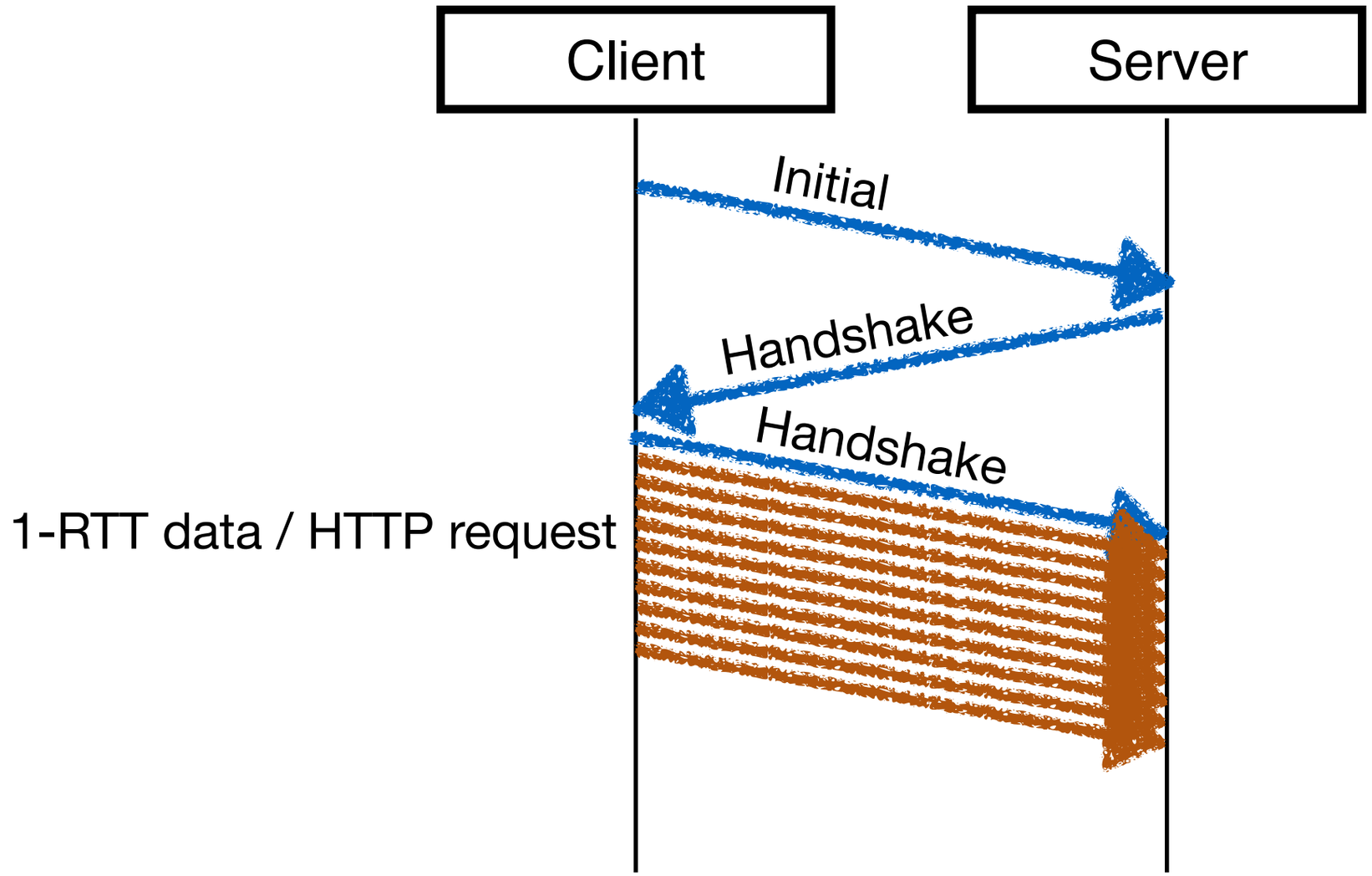
- Only Destination Connection ID or no Connection ID at all
- Payload is always encrypted (K bit indications key phase for decryption)

QUIC Invariants

- Basically everything can change between QUIC version, except a few invariants that are needed for version negotiation!

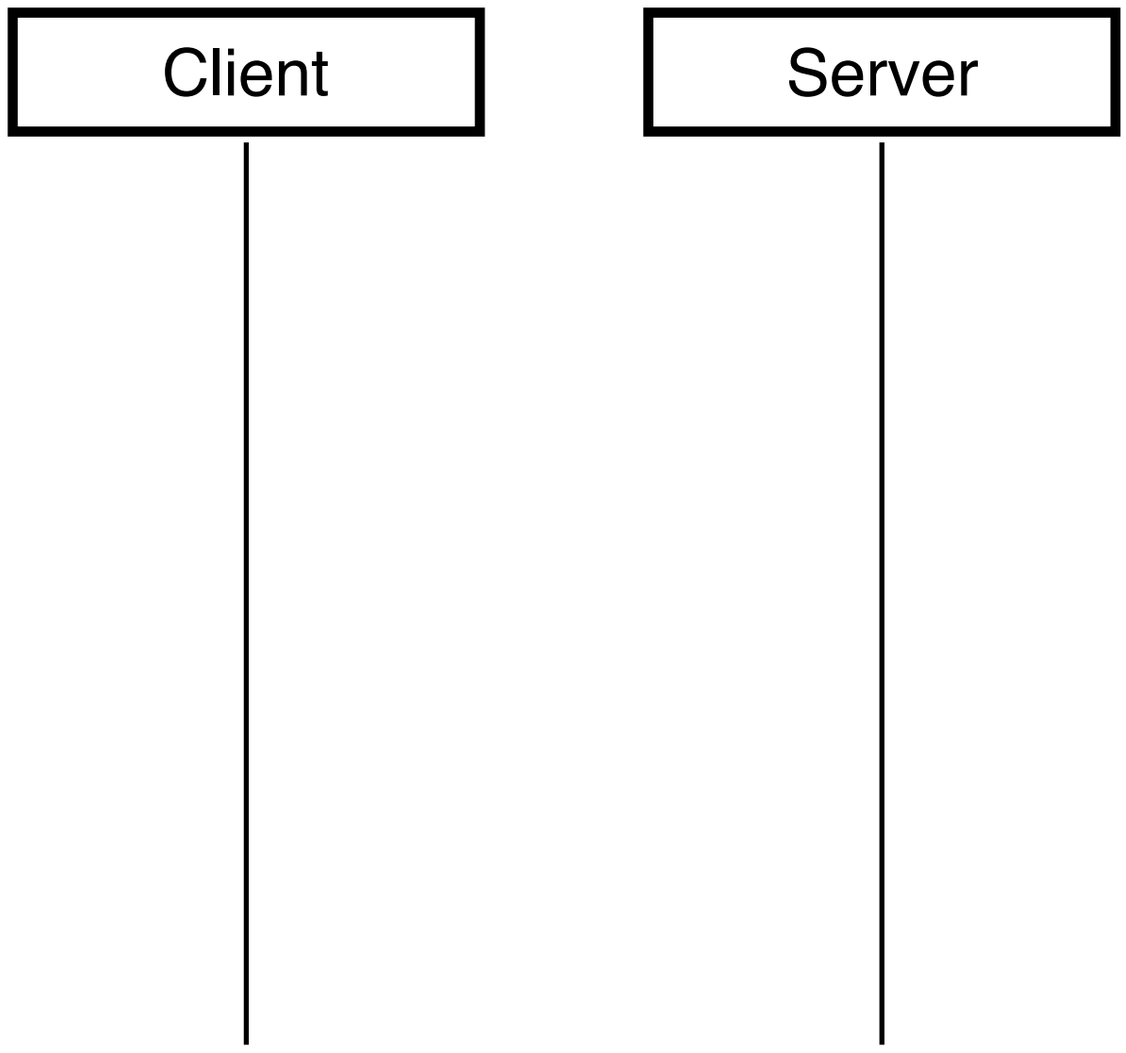
- Only to make version negotiation work, between different versions the following things need to remain the same:
  - the location and meaning of the header form flag,
  - the location and meaning of the Connection ID flag in the short header,
  - the location and size of the Connection ID field in both header forms,
  - the location and size and meaning of the Version field in long headers, and
  - the whole version negotiation packet.

QUIC Handshake



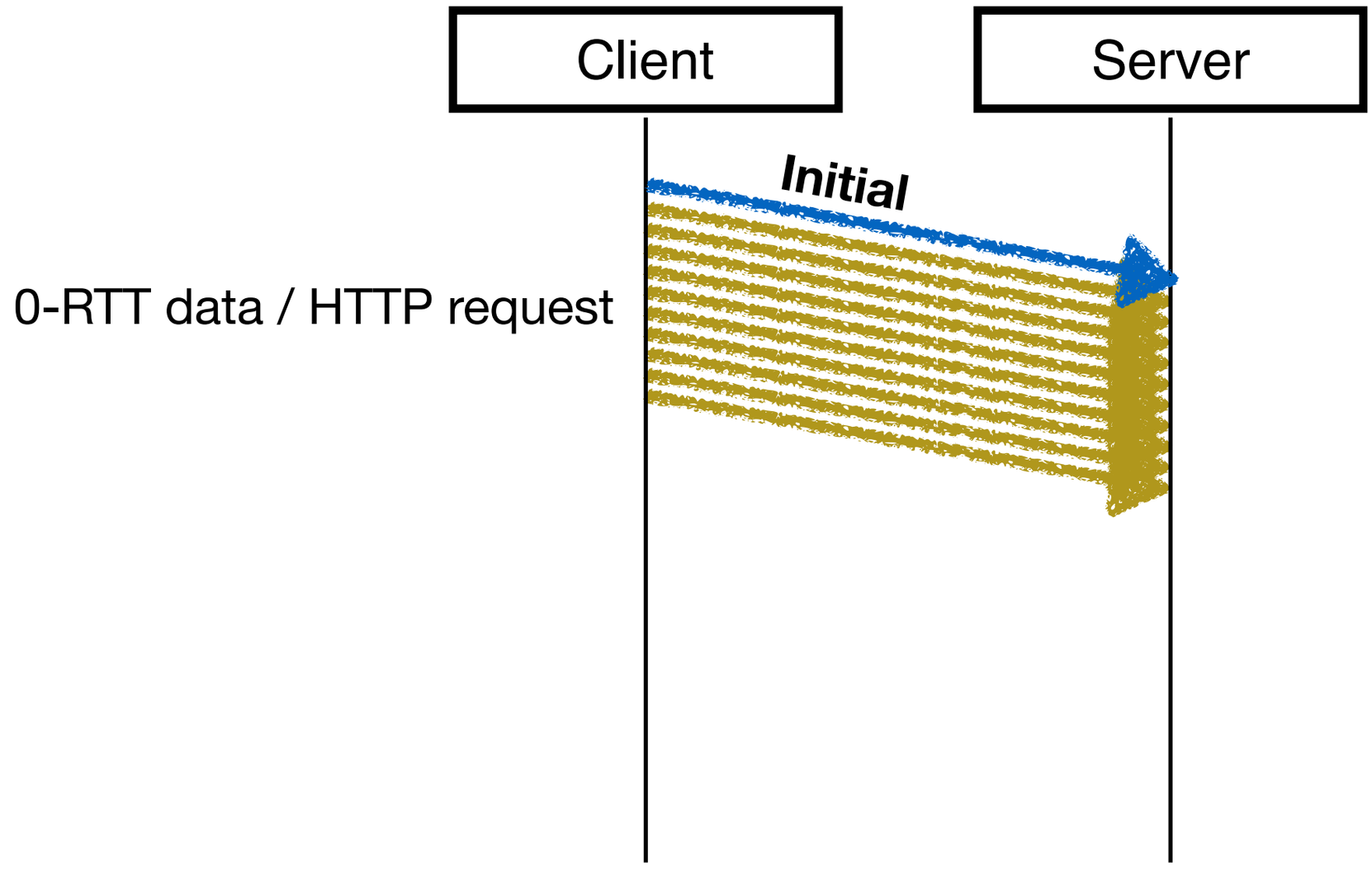
- Copy of the Client Initial and Server Cleartext are included in the encrypted part to verify content

QUIC Handshake — 0-RTT Session Resumption



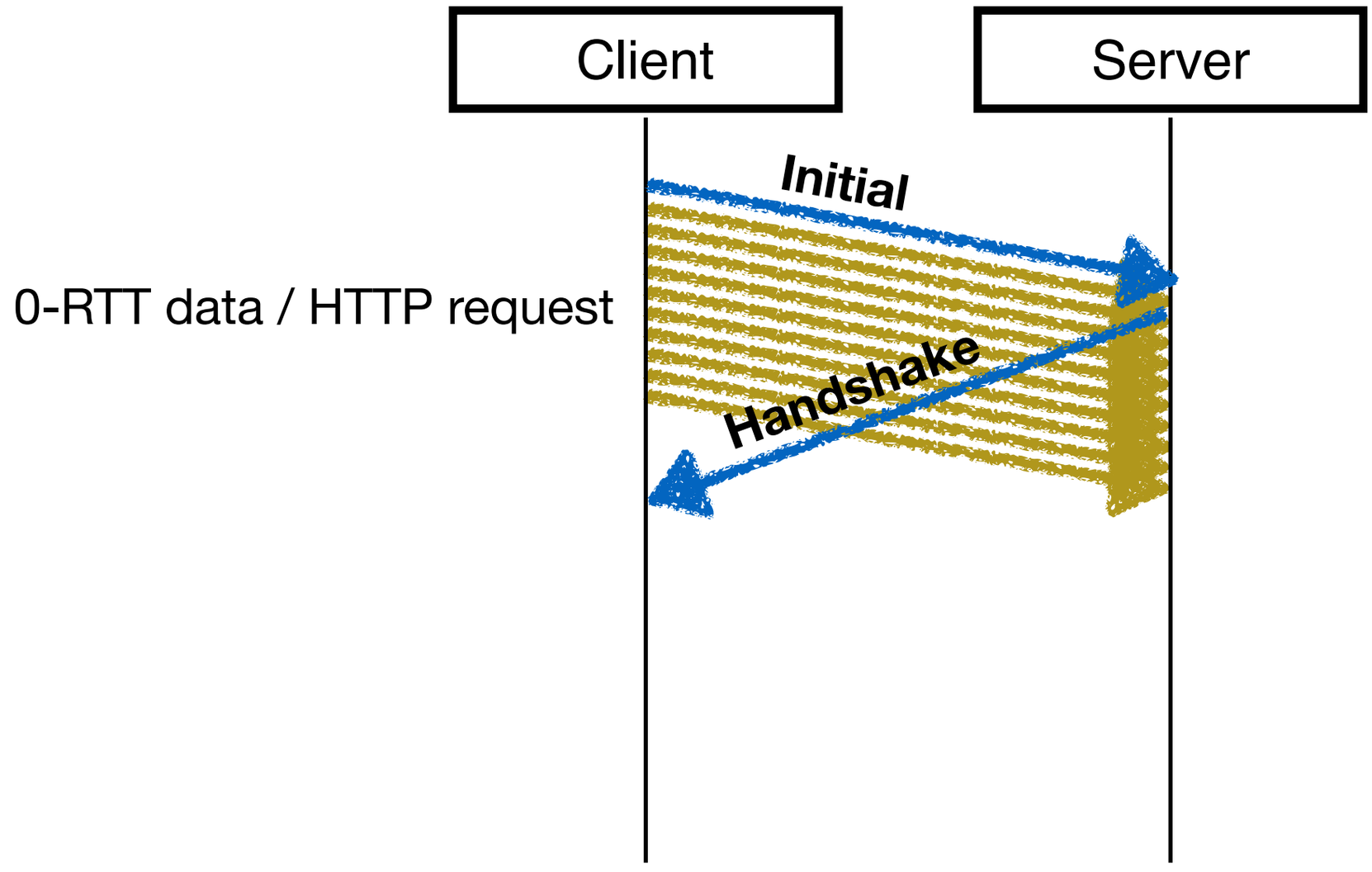
- Client can send an initial window of data (10 packets) together with Client Initial

QUIC Handshake — 0-RTT Session Resumption



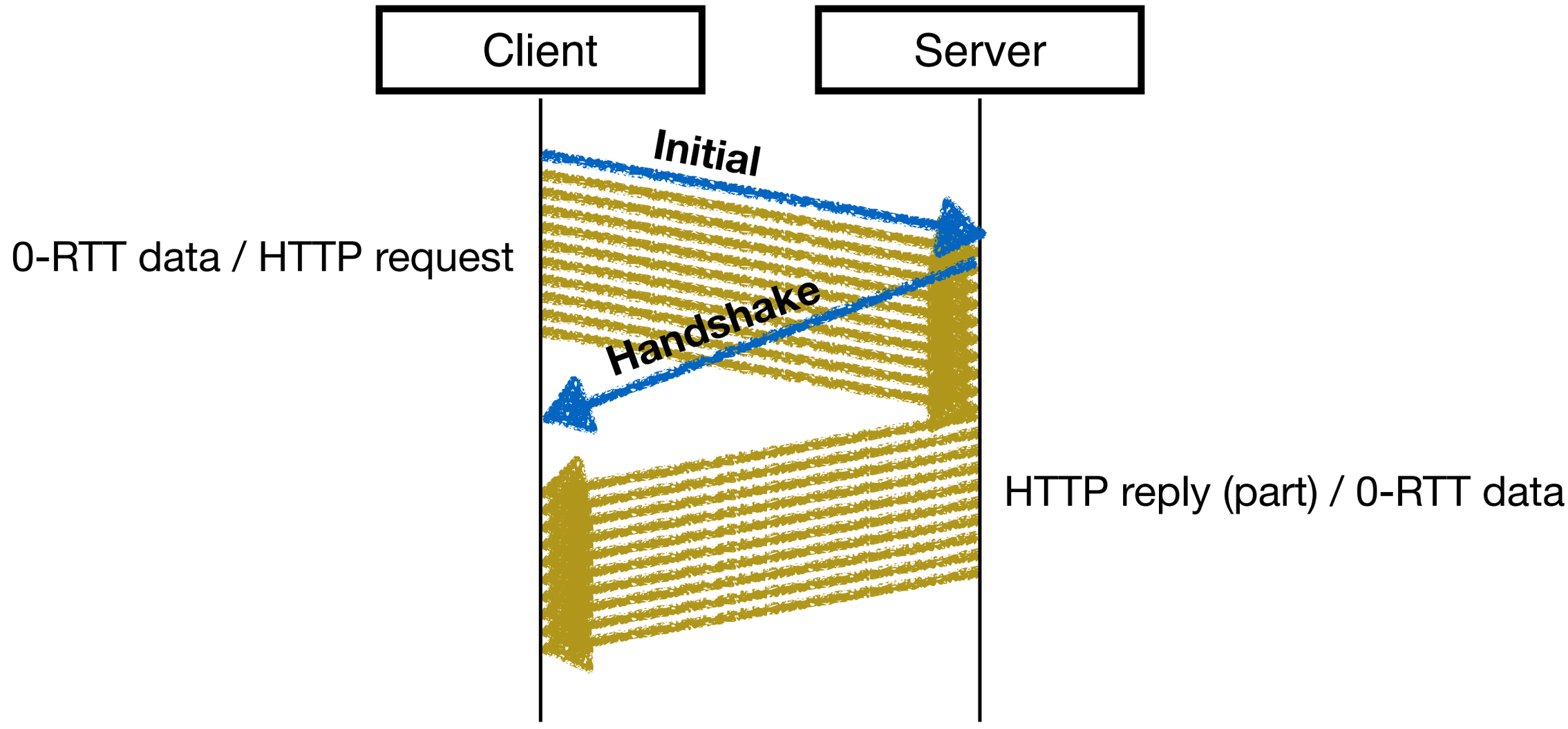
- Client can send an initial window of data (10 packets) together with Client Initial

QUIC Handshake — 0-RTT Session Resumption



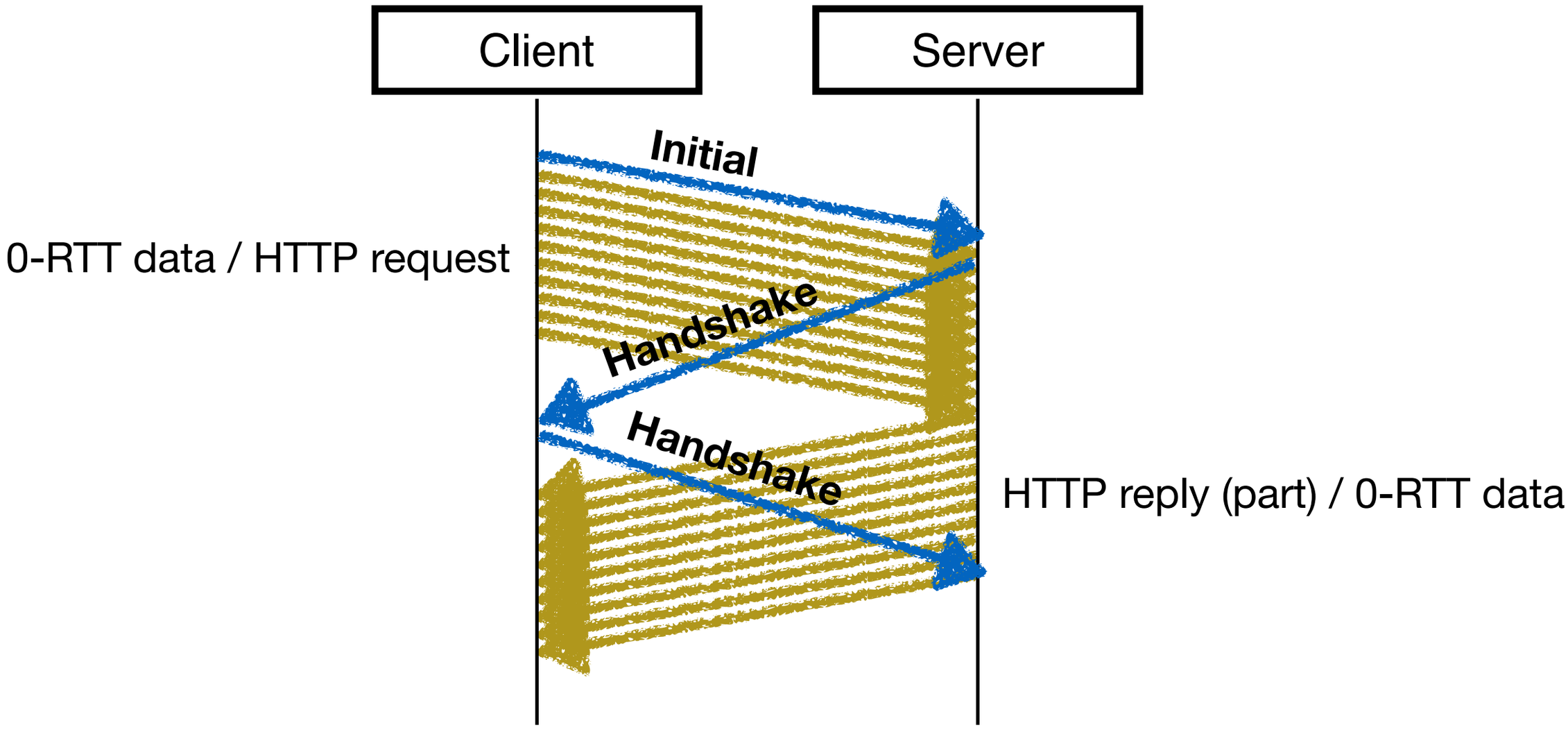
- Client can send an initial window of data (10 packets) together with Client Initial

QUIC Handshake — 0-RTT Session Resumption



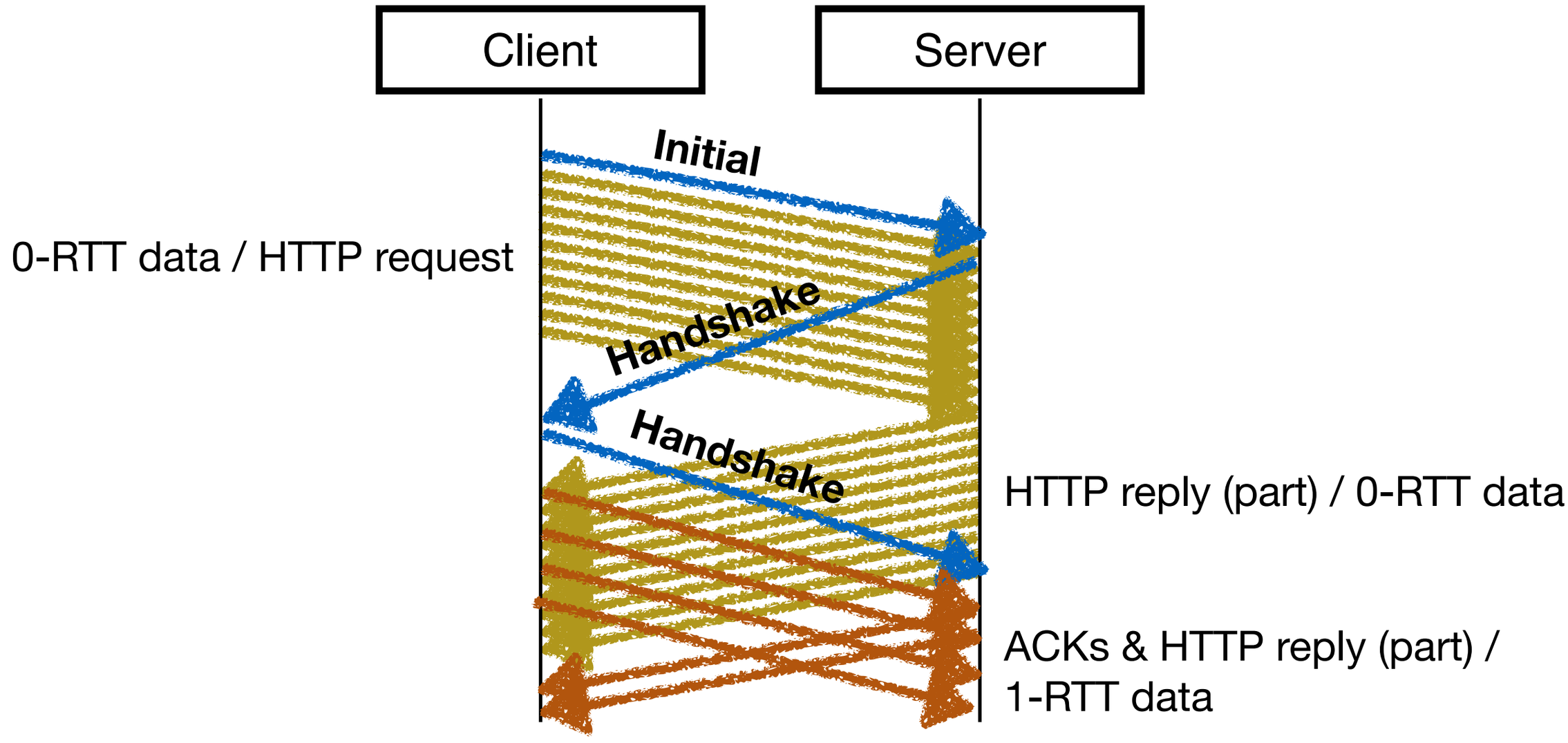
- Client can send an initial window of data (10 packets) together with Client Initial

QUIC Handshake — 0-RTT Session Resumption



- Client can send an initial window of data (10 packets) together with Client Initial

QUIC Handshake — 0-RTT Session Resumption

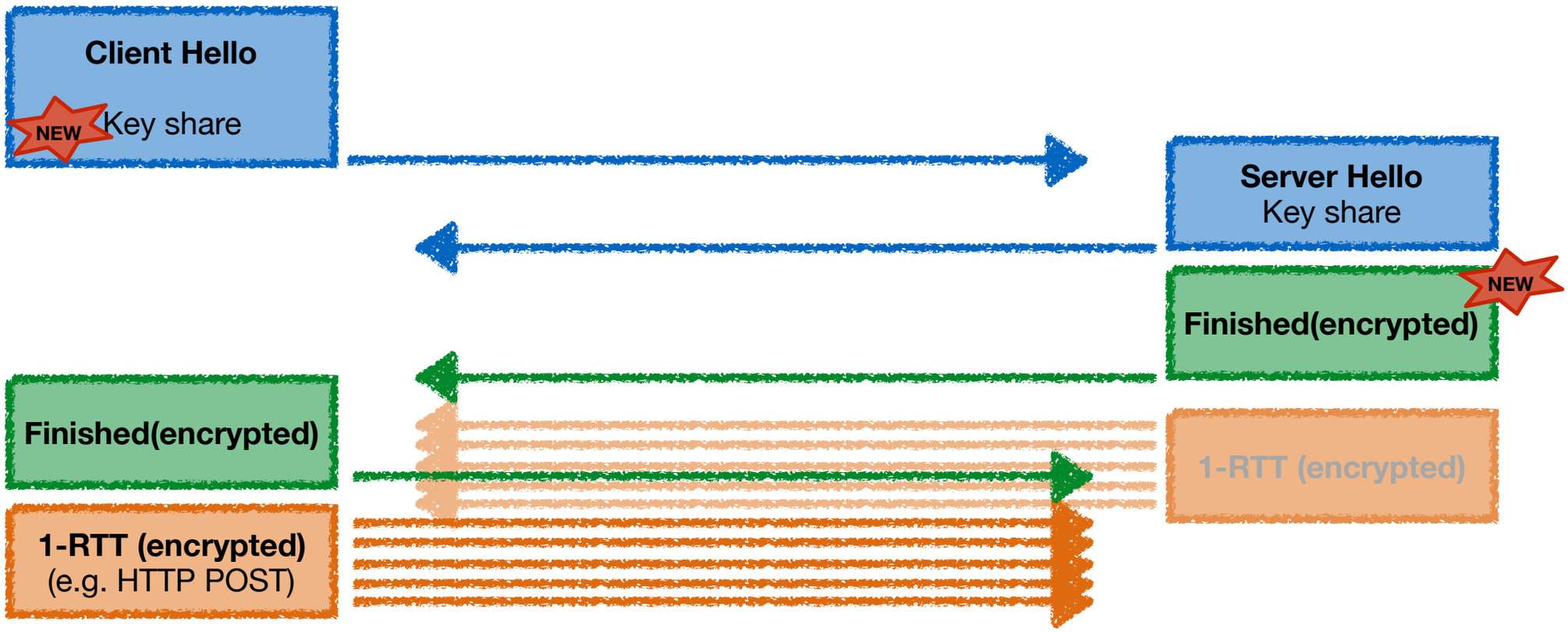


- Client can send an initial window of data (10 packets) together with Client Initial

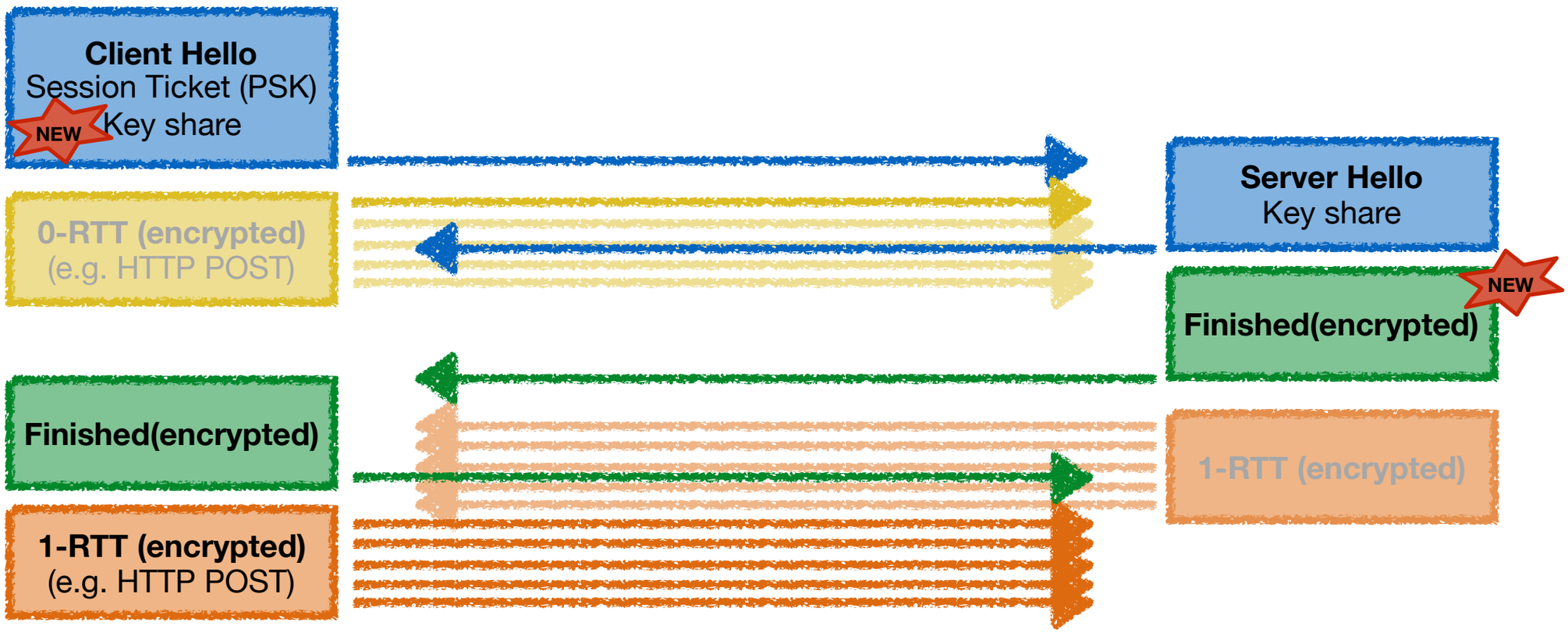
TLS1.3 - Features

- Use of Authenticated Encryption with Associated Data (AEAD)
- Perfect Forward Secrecy
  - Static RSA and static Diffie-Hellman cipher suites removed
- All handshake messages after the ServerHello now encrypted
- New version negotiation mechanism based on extension due for (server and network) compatibility
- Session resumption in 0-RTT mode

TLS 1.3 Handshake



TLS 1.3 Handshake



- Session Tickets servers are stateless
  - 0-RTT PSK provides no Forward Secrecy if Session Ticket key is compromised
  - Replay attack requires idempotent data for 0-RTT

TLS1.3 Interception

- see also draft-camwinget-tls-use-cases

- Perfect Forward Secrecy
  - Keys cannot be shared apriori with middlebox
  - (see also draft-green-tls-static-dh-in-tls13)
- Encrypted Server Certificate
  - Server Identity hidden (as Server Name Identification (SNI) might not match)
- Downgrade Protection
  - detects stripping of "supported_versions" extension

- Certificate pinning
  - see also draft-sheffer-tls-pinning-ticket

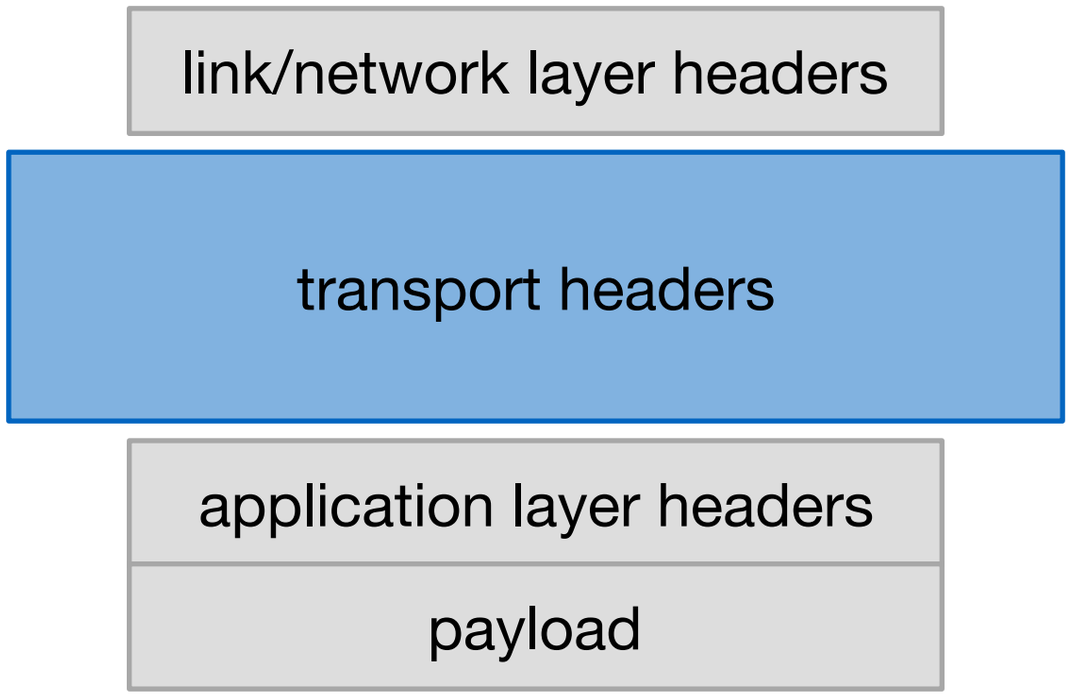
QUIC wire image — Measurements and Monitoring

- **Round-Trip Time (RTT)** can be estimated during the handshake
  - ➡ No easy way to correlate two packets in each direction during the rest of the connection

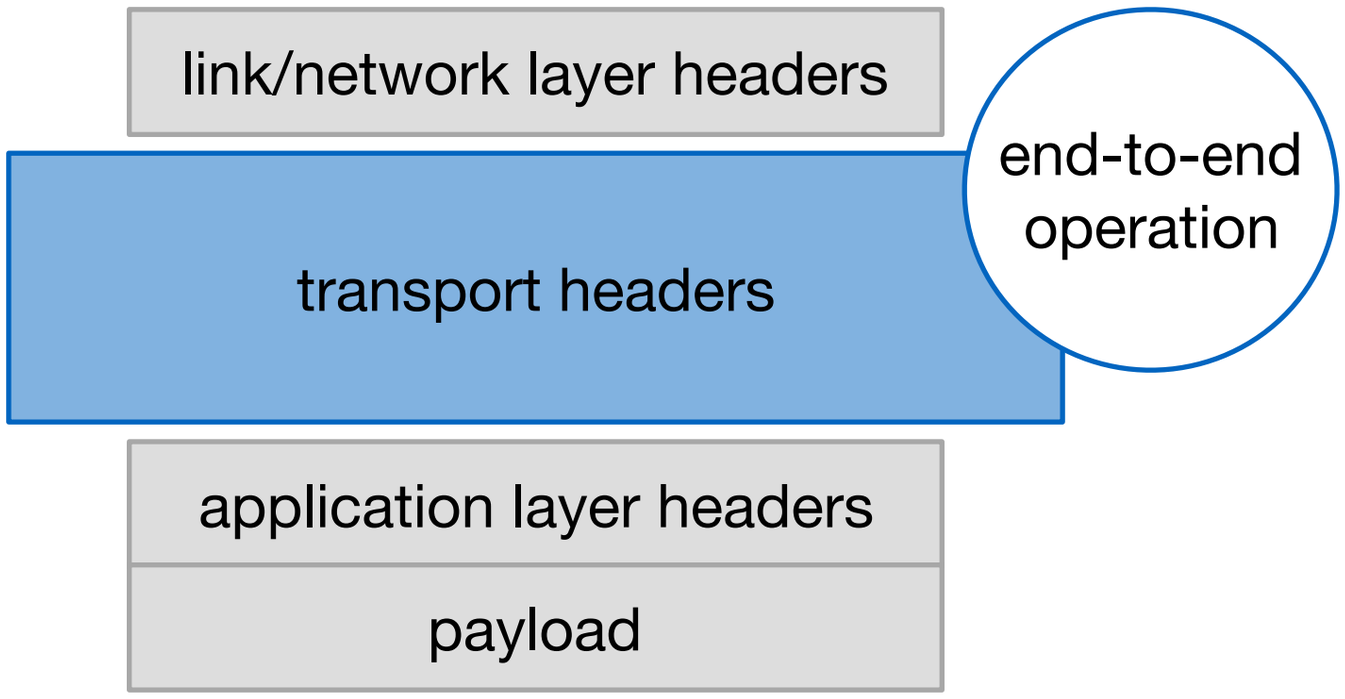
- Packet Number is now encrypted and cannot be utilized for measurement anymore

- Retransmissions and ECN congestion indications are not visible to the path (to estimate whole-path congestion)

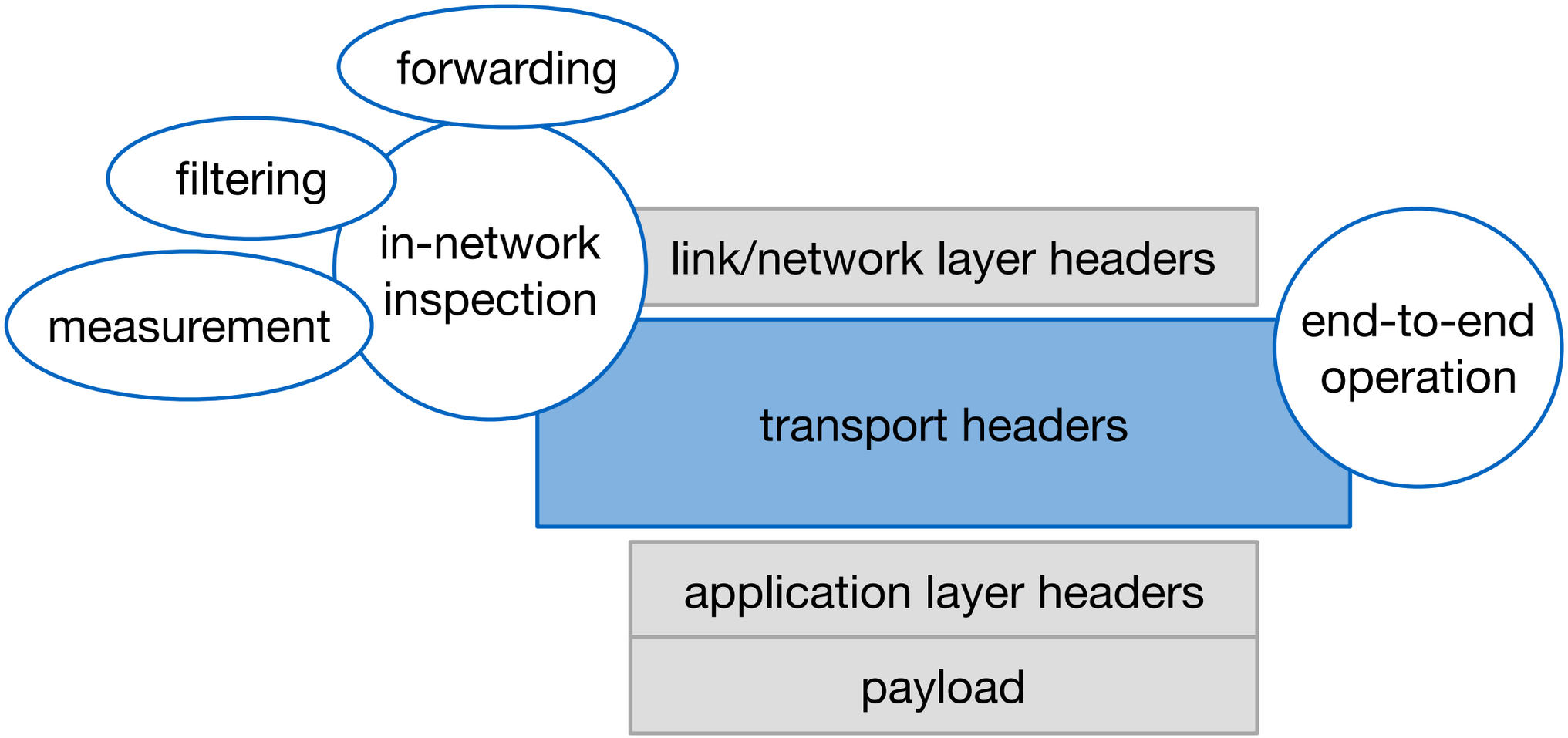
Transport protocol design, 1990s style



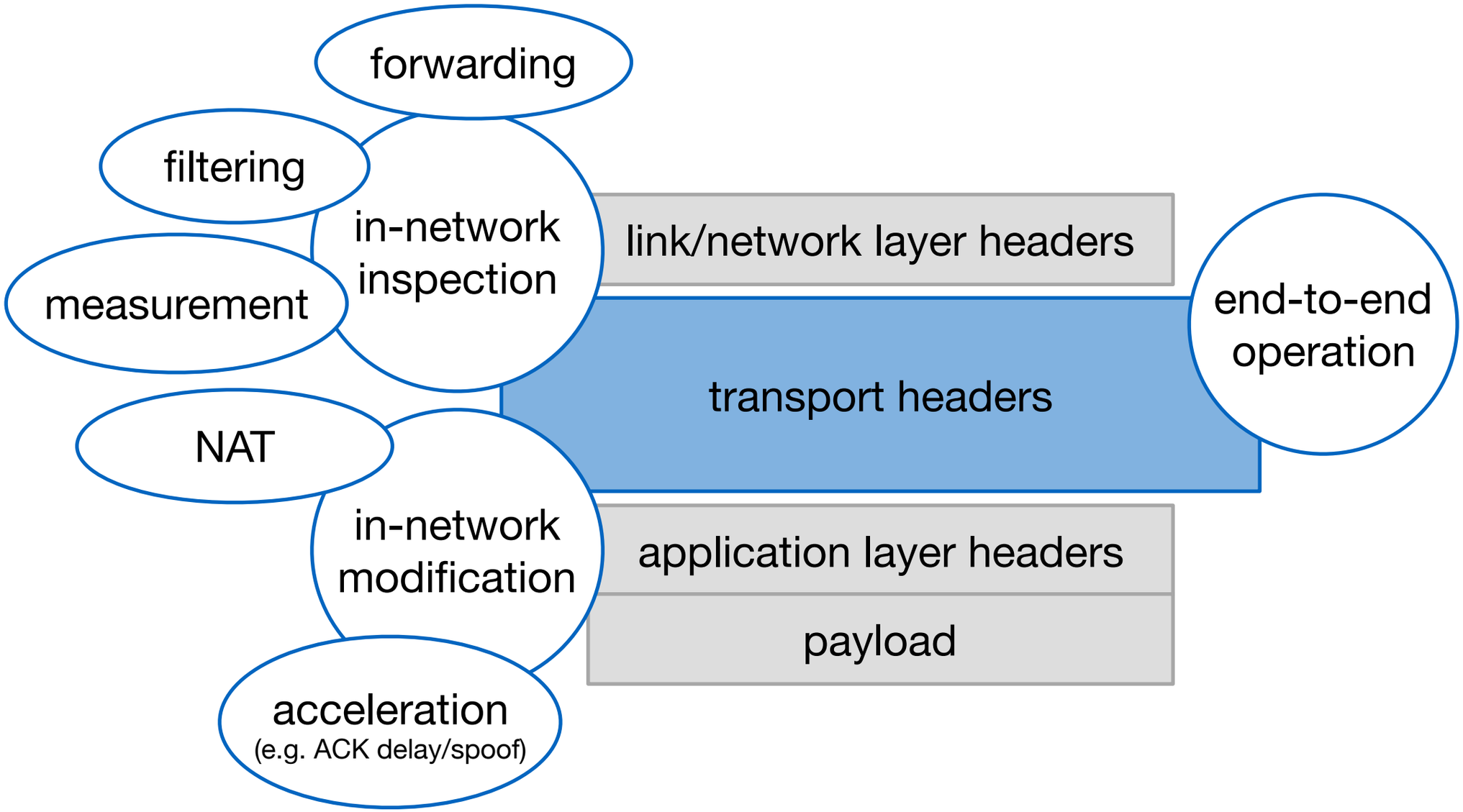
Transport protocol design, 1990s style



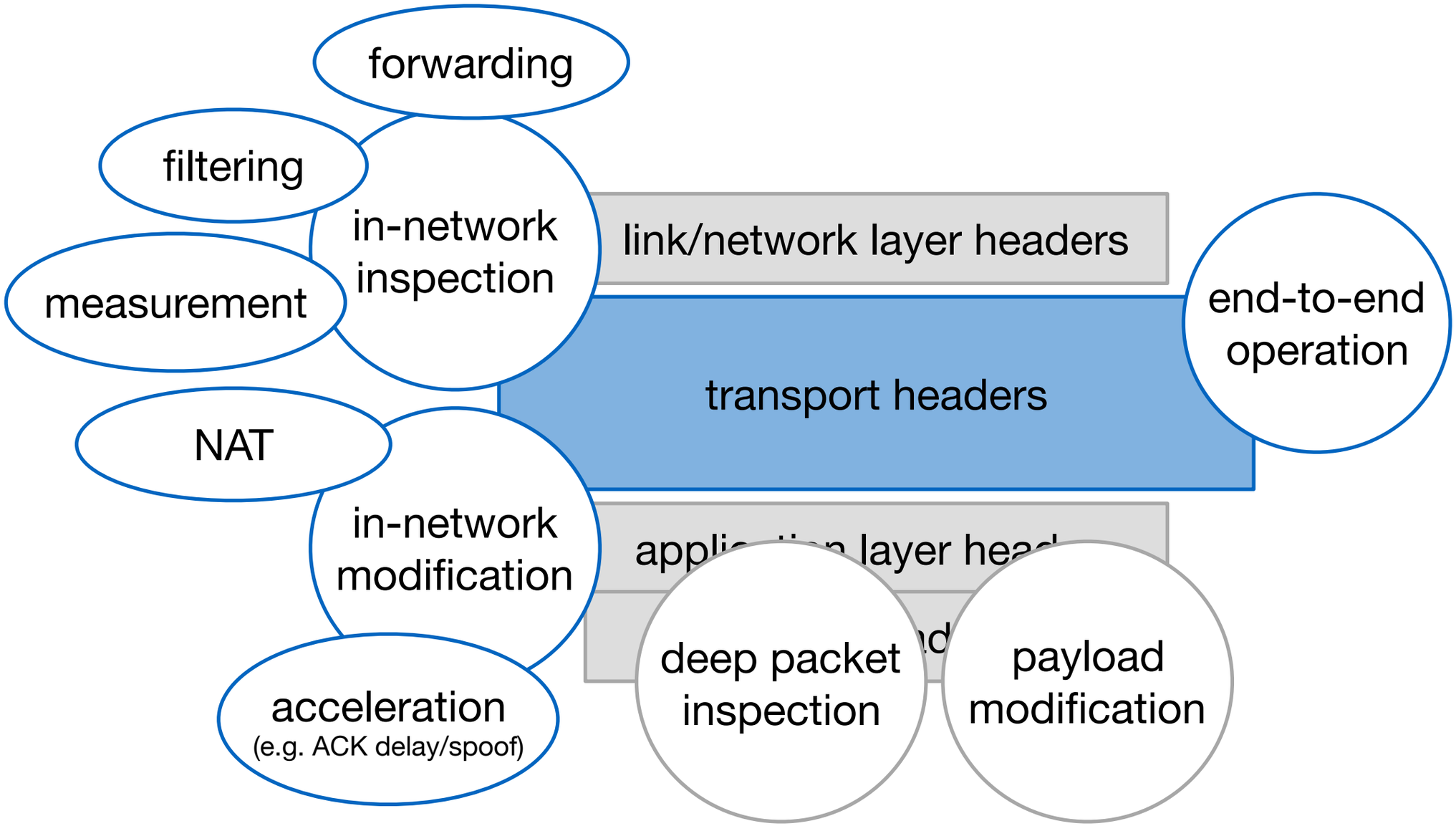
Transport protocol design, 1990s style



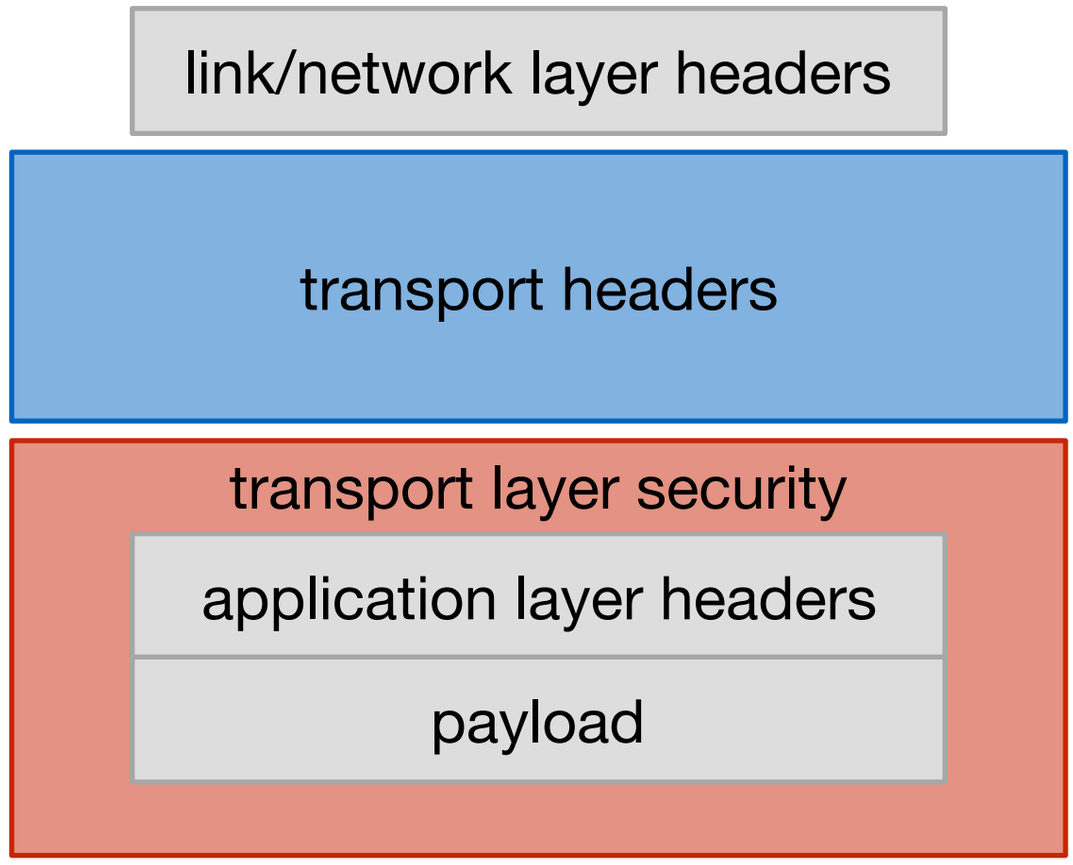
Transport protocol design, 1990s style



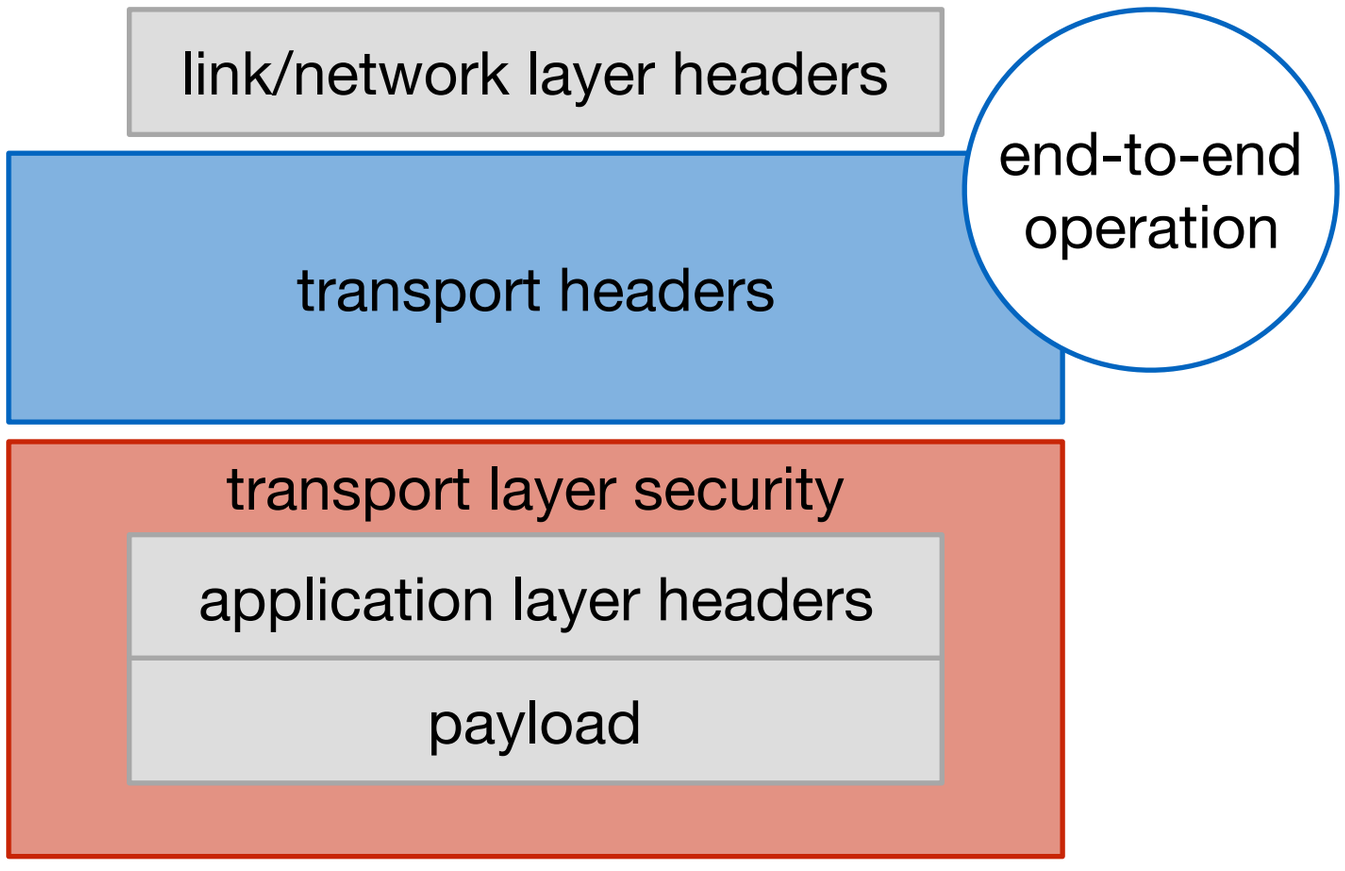
Transport protocol design, 1990s style



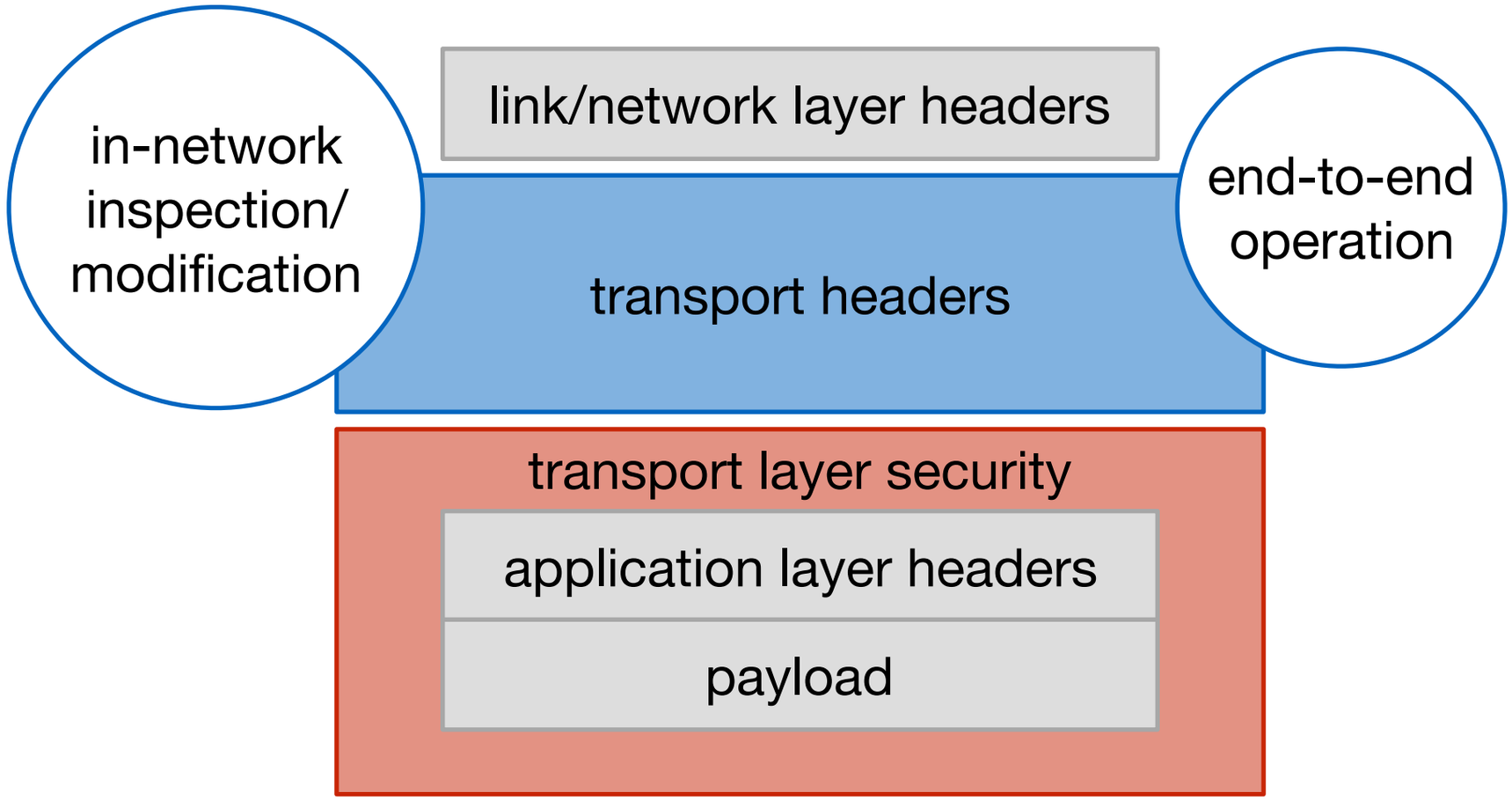
Transport protocol design, now with security!



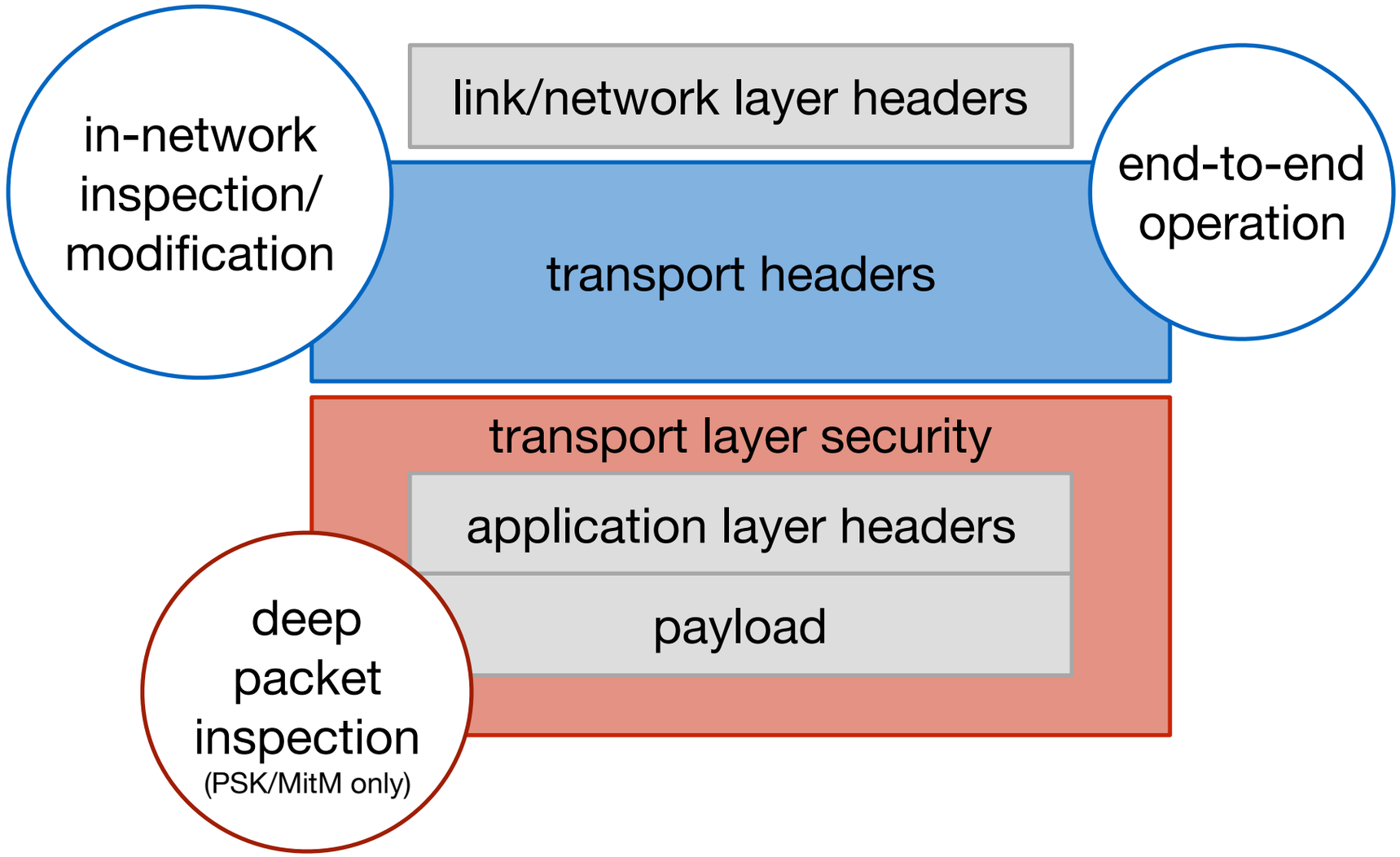
Transport protocol design, now with security!



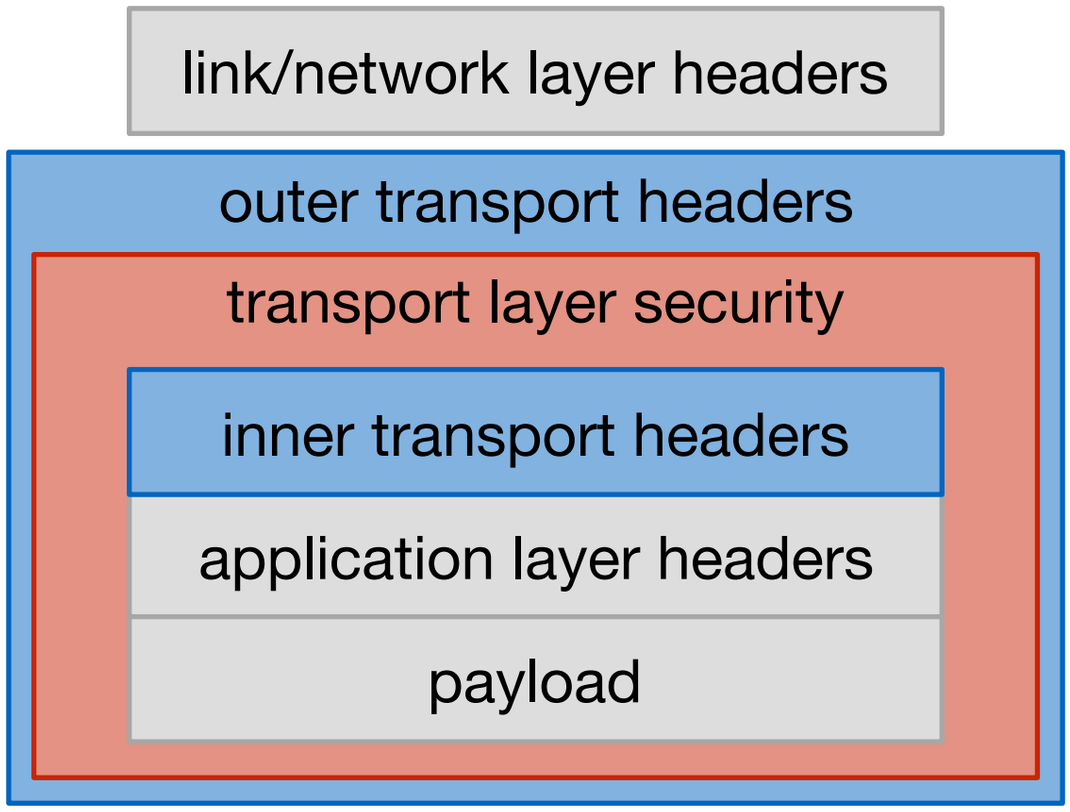
Transport protocol design, now with security!



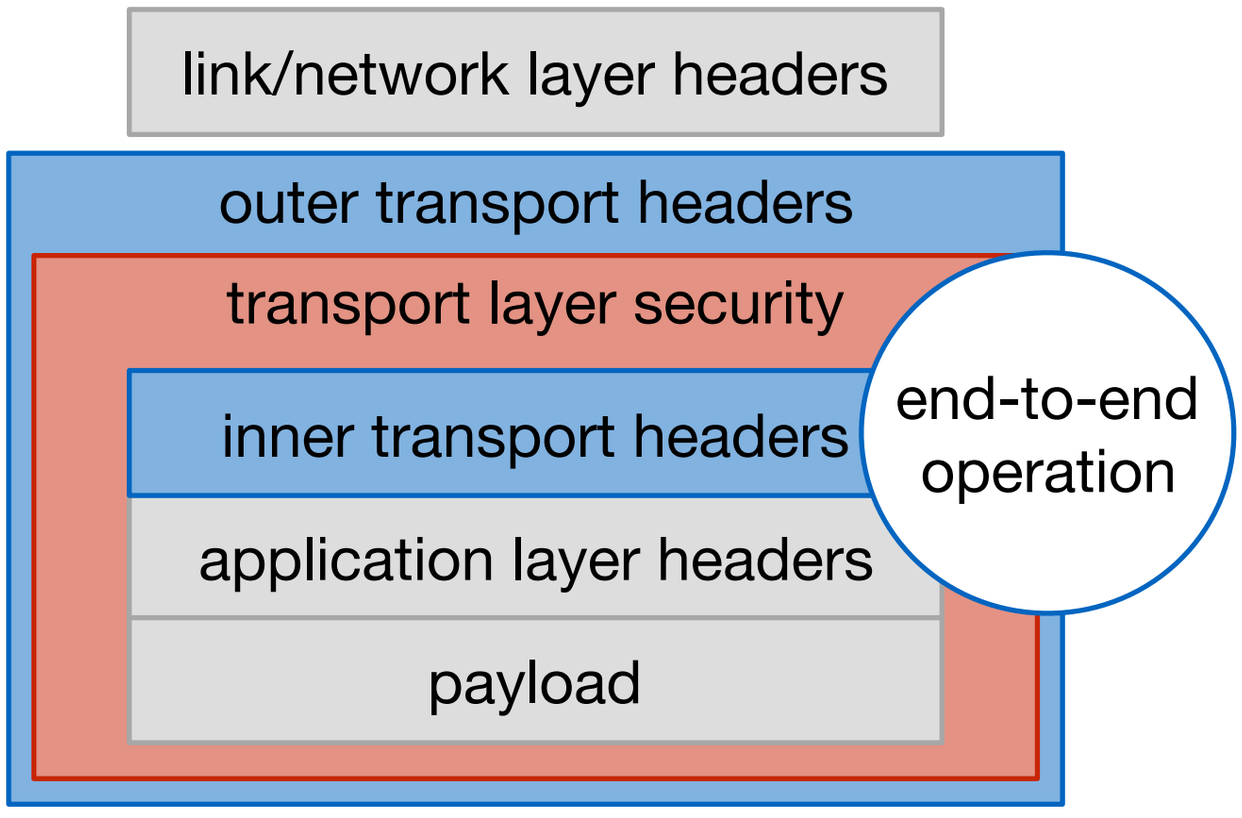
Transport protocol design, now with security!



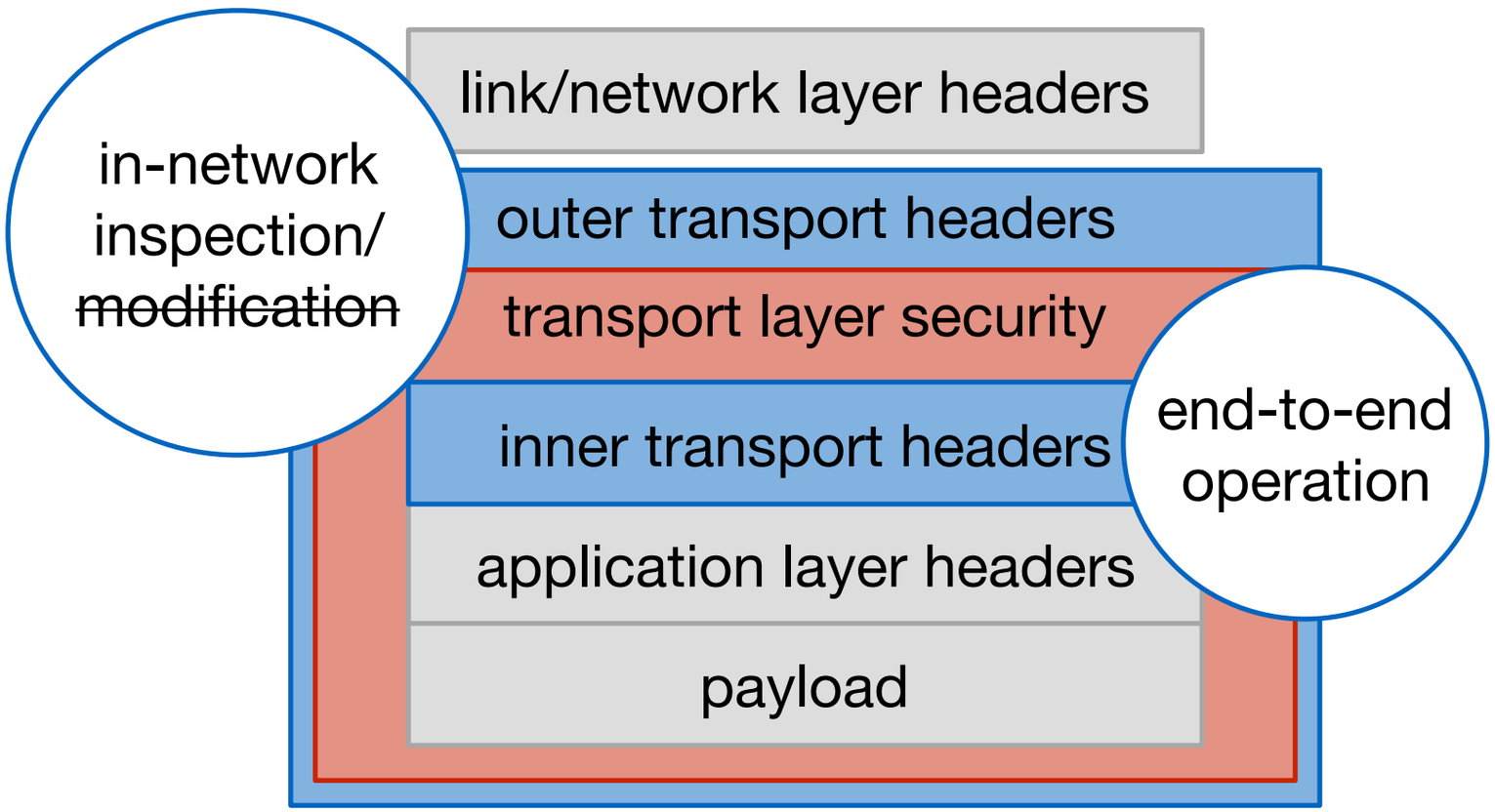
Encrypted transport protocol design: introducing the wire image



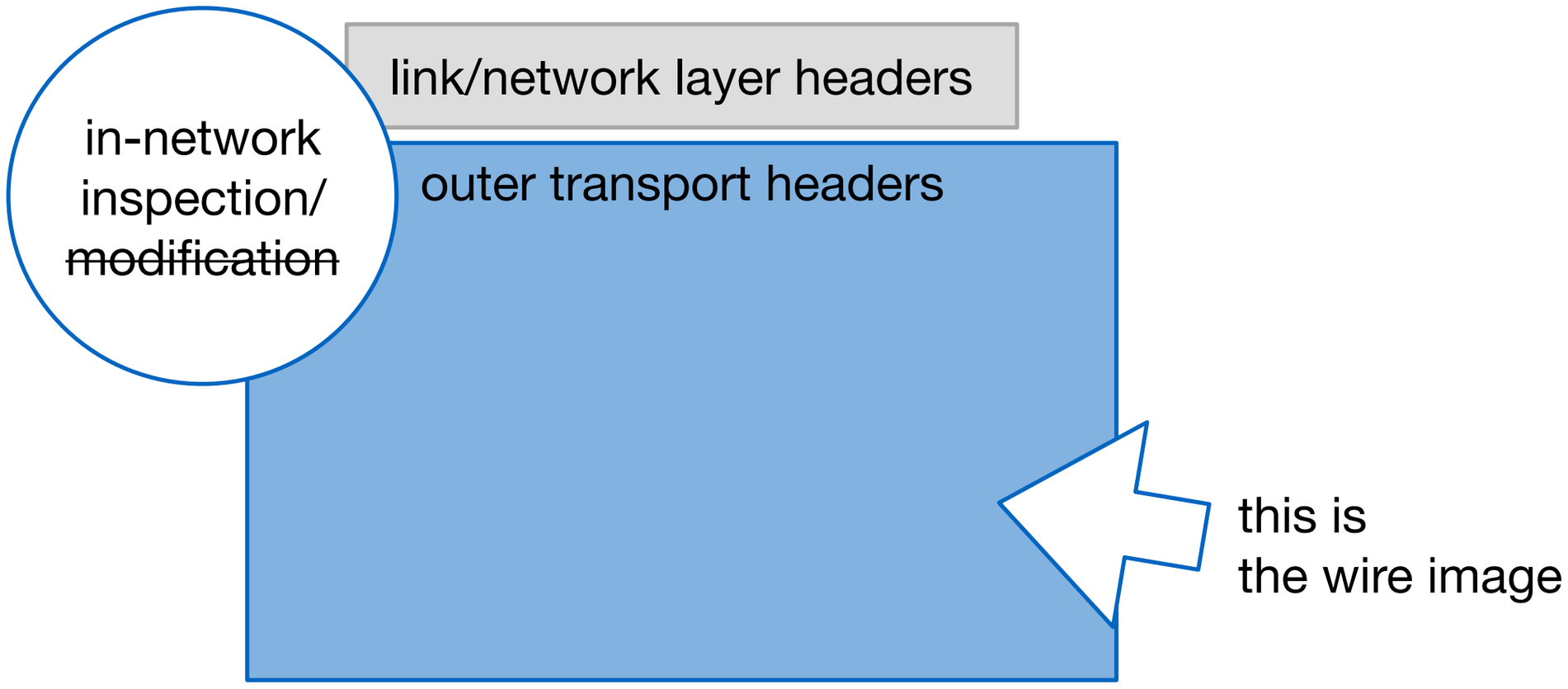
Encrypted transport protocol design: introducing the wire image



Encrypted transport protocol design: introducing the wire image



Encrypted transport protocol design: introducing the wire image



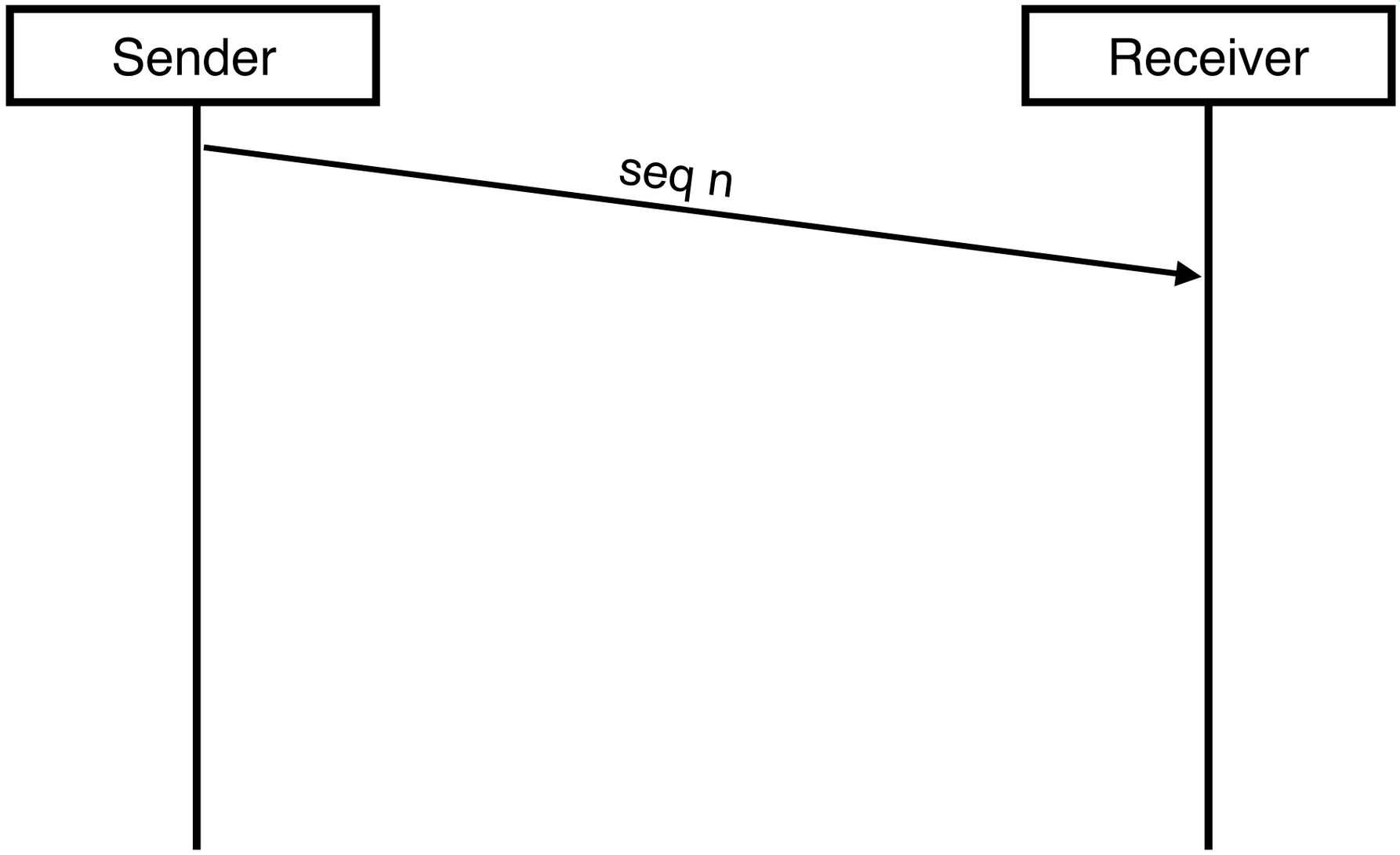
The Wire Image: Three levels of accessibility

- Unprotected (as TCP today): all bits can be seen at all points along the path, and can be modified without endpoint knowledge.
  - Supports manipulation of transport internals.
- Integrity-protected (e.g. QUIC outer header): all bits can be seen at all points along the path, but modification is endpoint detectable (and leads to packet rejection).
- Encrypted (e.g. QUIC frame headers): no bits can be interpreted, modification leads to corruption and rejection.

RTT estimation with TCP

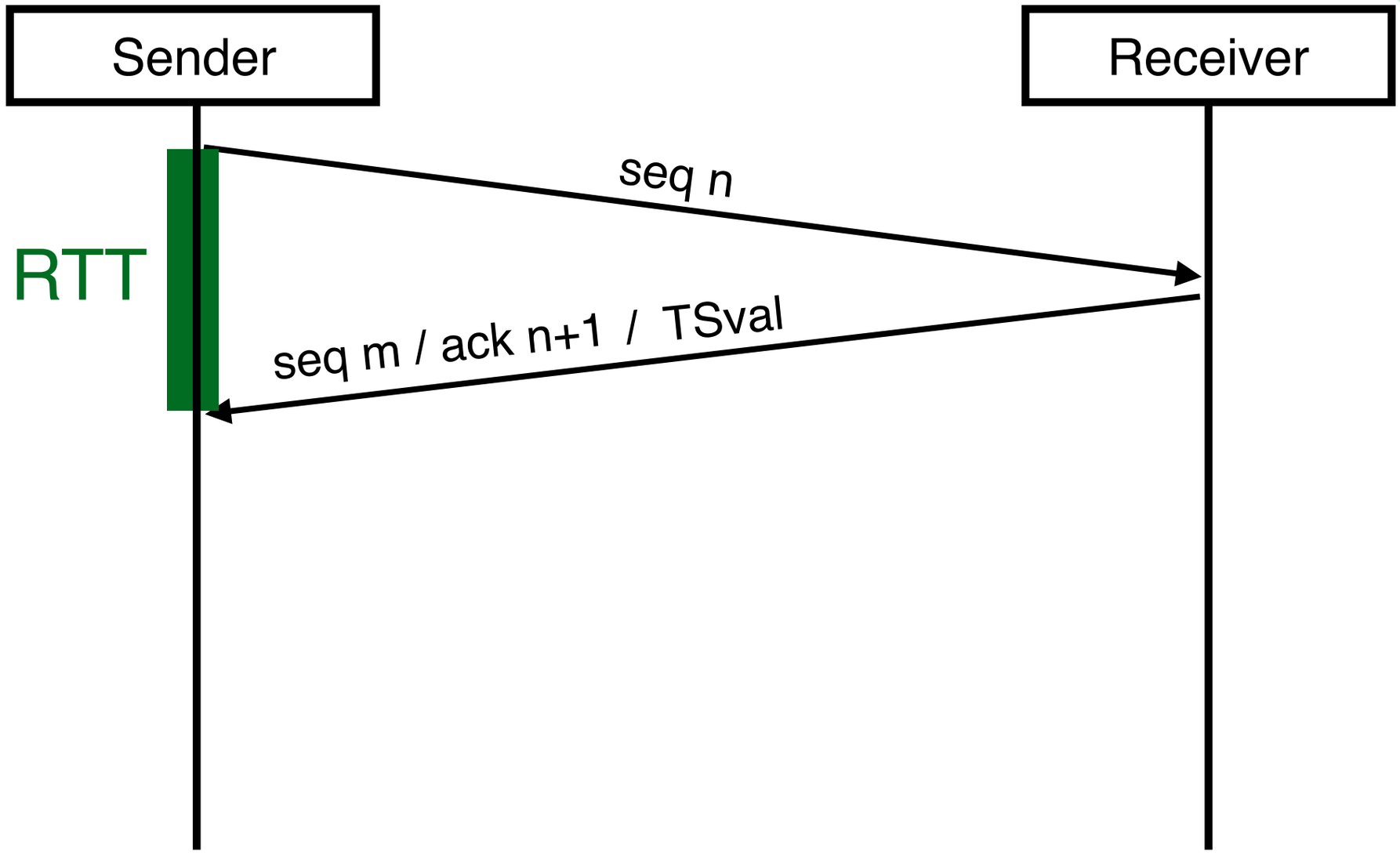
- Use of SEQ# and ACK# and/or TCP Timestamp Option

RTT estimation with TCP



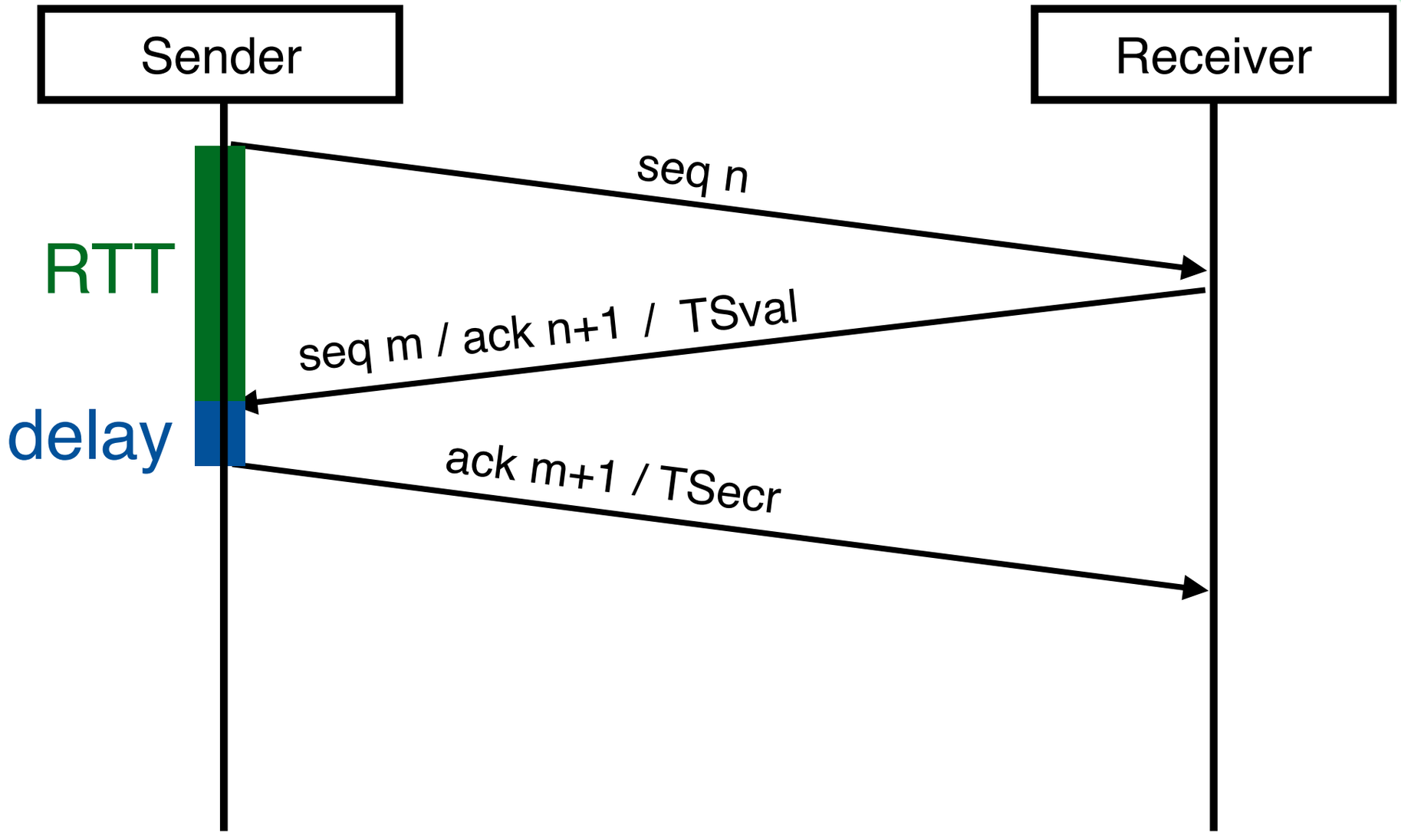
- Use of SEQ# and ACK# and/or TCP Timestamp Option

RTT estimation with TCP



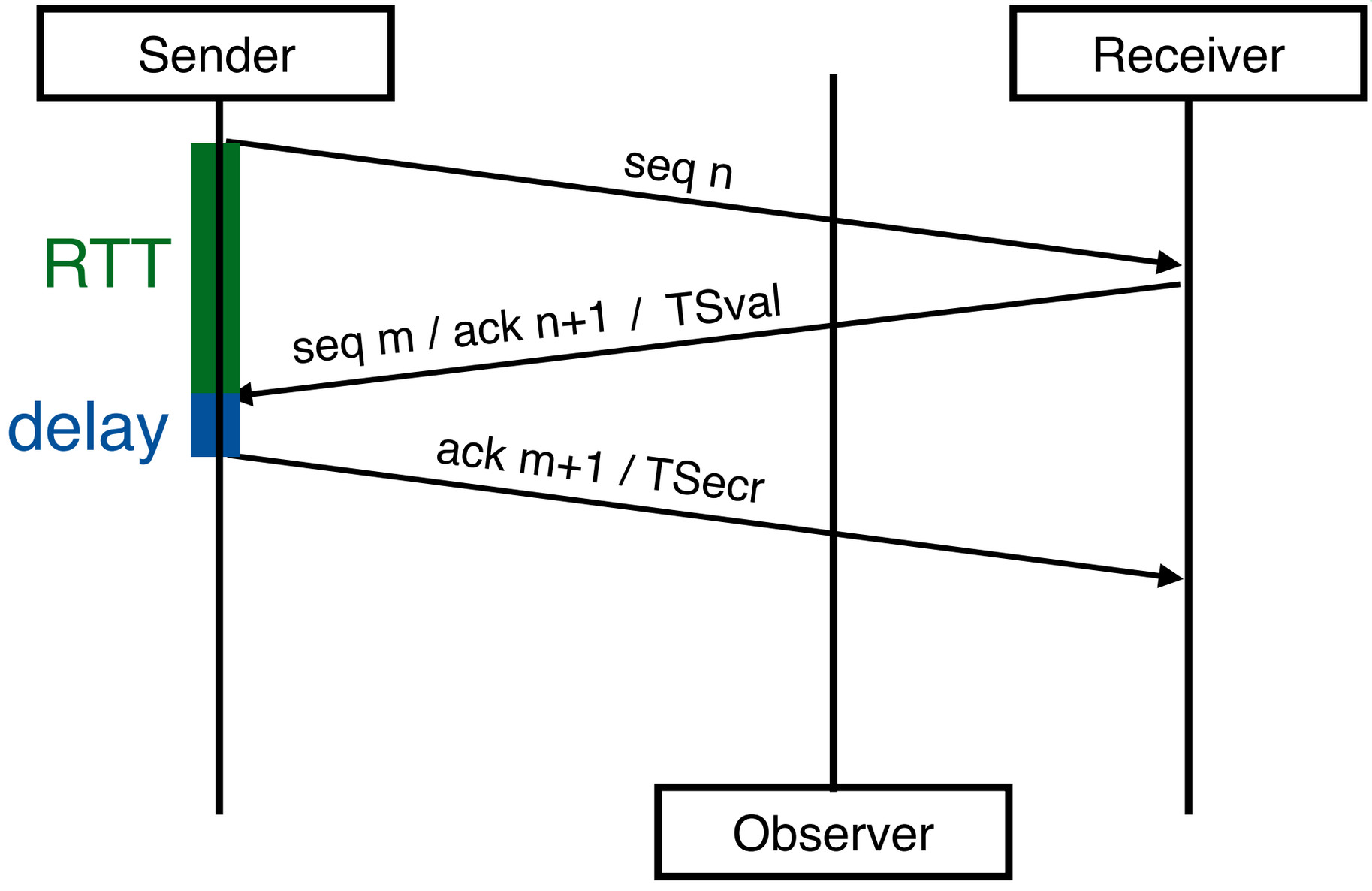
- Use of SEQ# and ACK# and/or TCP Timestamp Option

RTT estimation with TCP



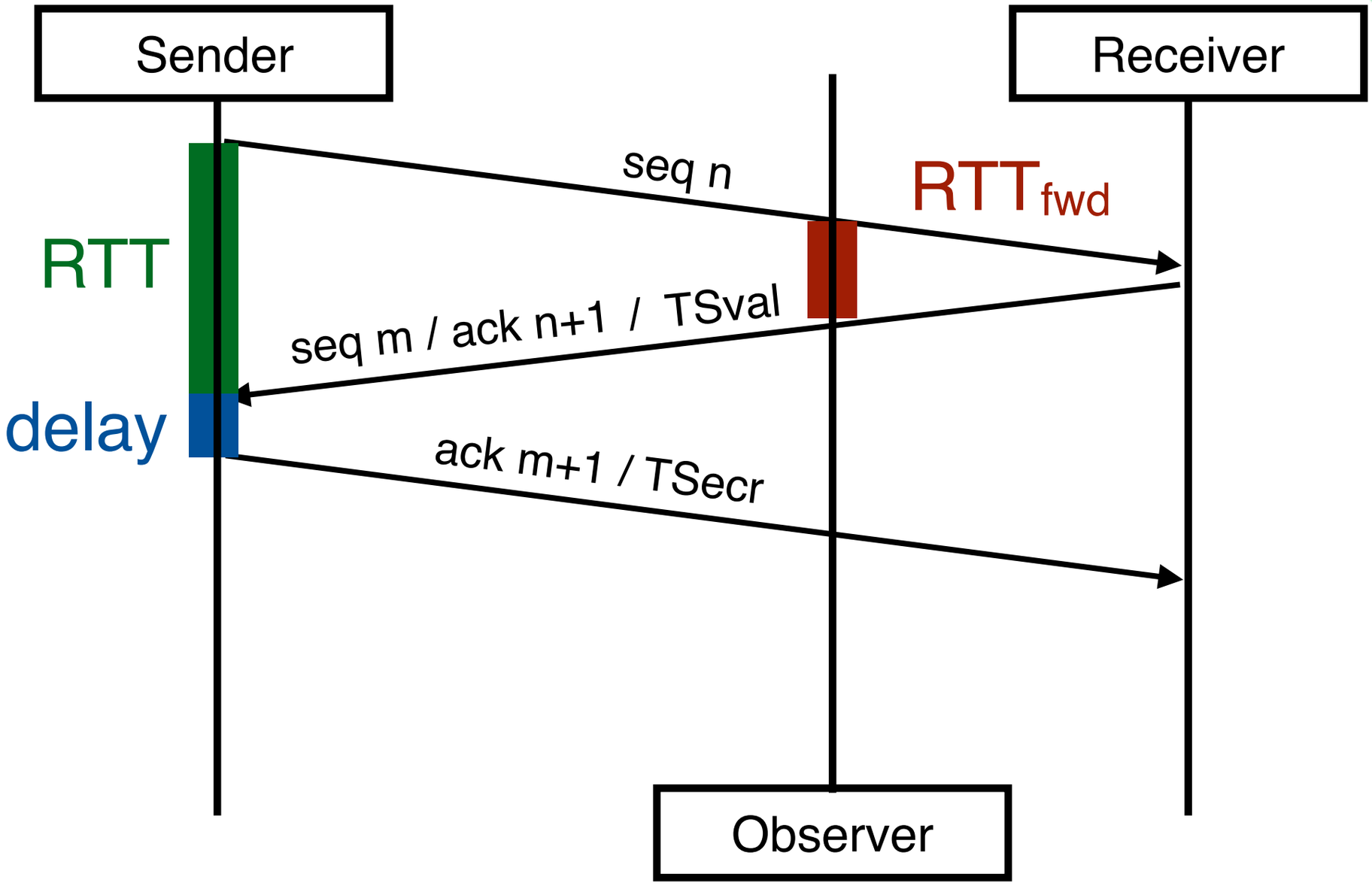
- Use of SEQ# and ACK# and/or TCP Timestamp Option

RTT estimation with TCP



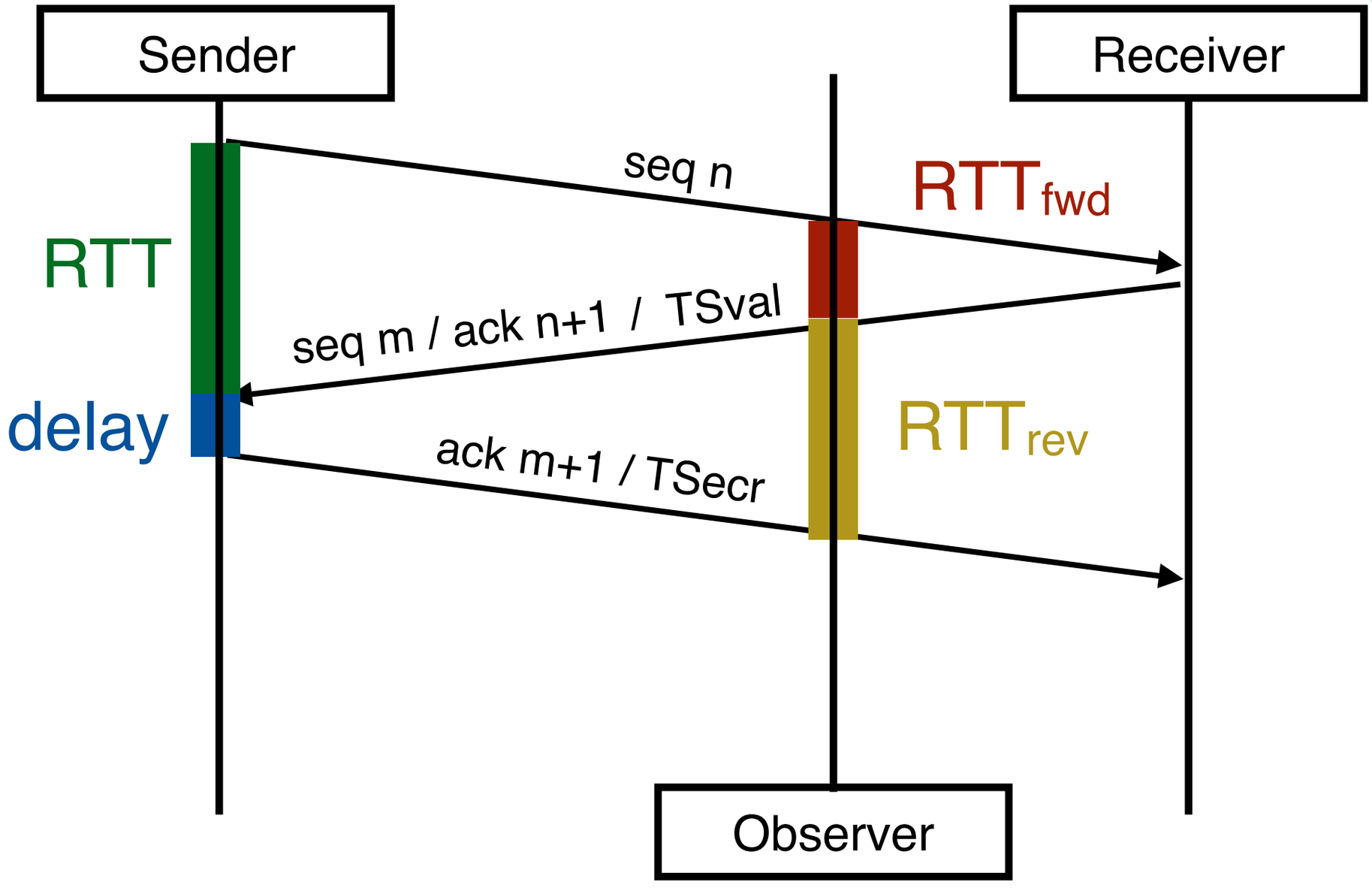
- Use of SEQ# and ACK# and/or TCP Timestamp Option

RTT estimation with TCP



- Use of SEQ# and ACK# and/or TCP Timestamp Option

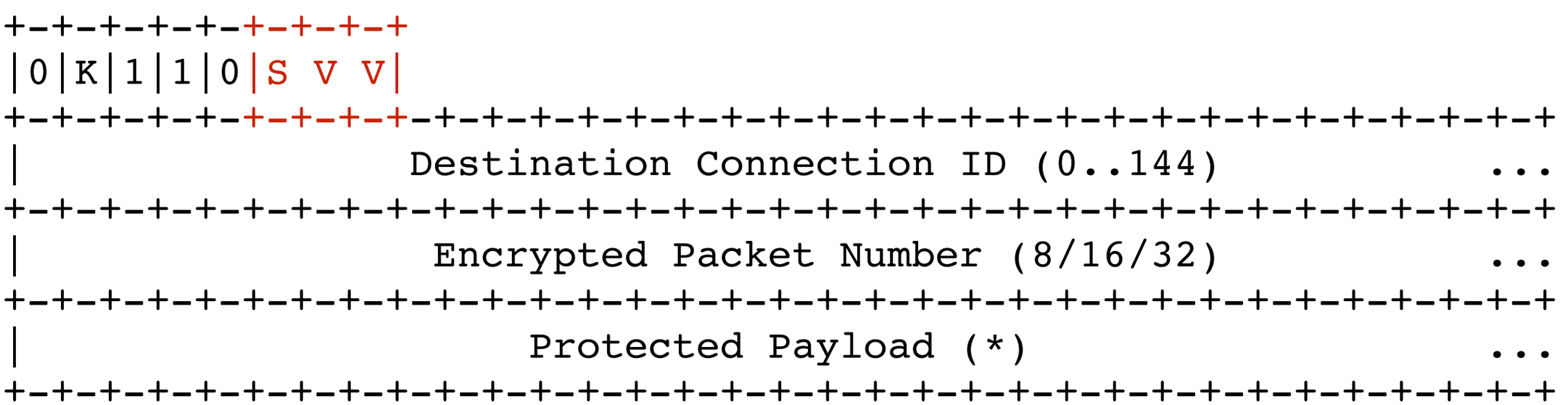
RTT estimation with TCP



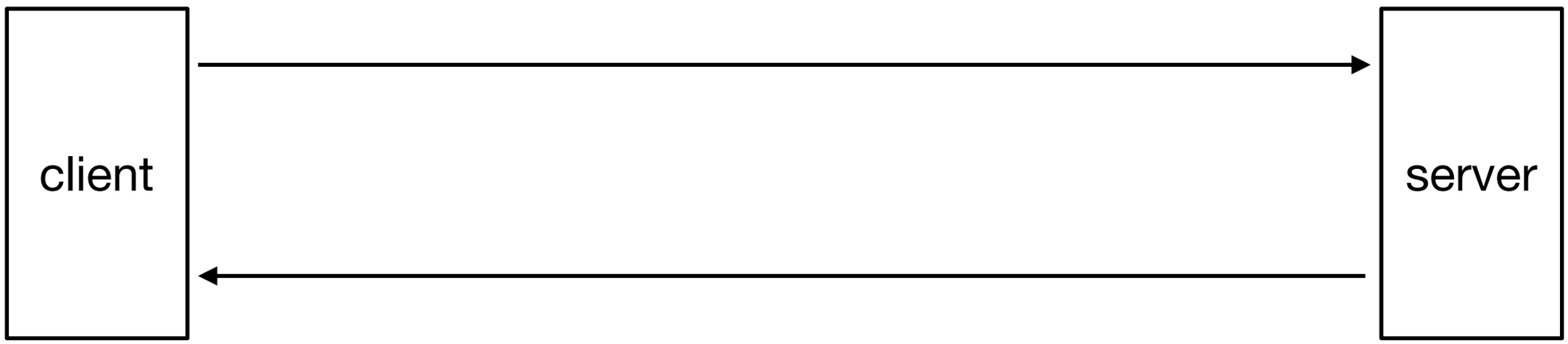
- Use of SEQ# and ACK# and/or TCP Timestamp Option

Replacing TCP RTT Measurement in the QUIC Wire Image: the Spin Bit

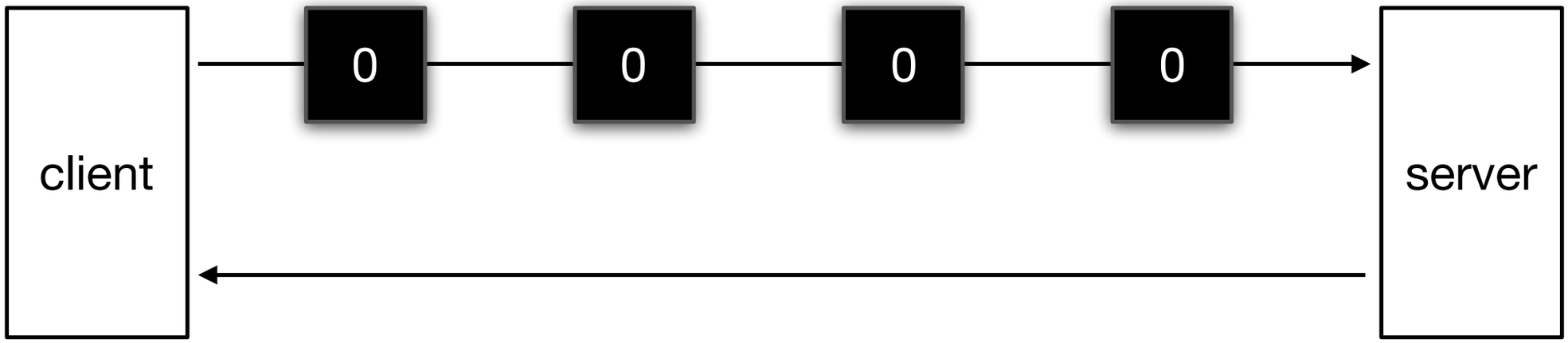
- Proposal: take a bit from QUIC short header type field and make it spin
- Server sets last spin it saw on each packet it sends
- Client sets ~(last spin it saw) on each packet it sends
- Creates a square-wave with period == RTT (when sender not app-limited)



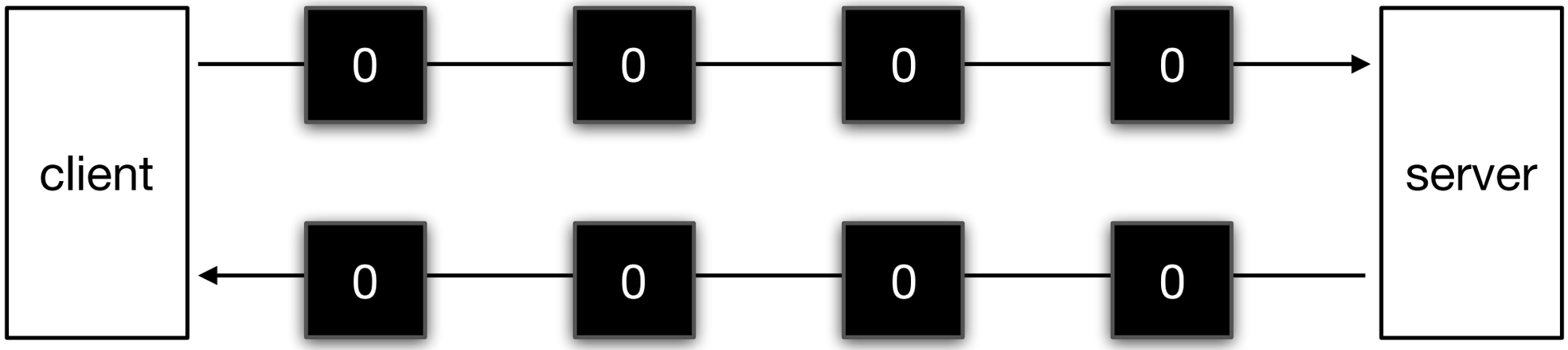
How does it work?



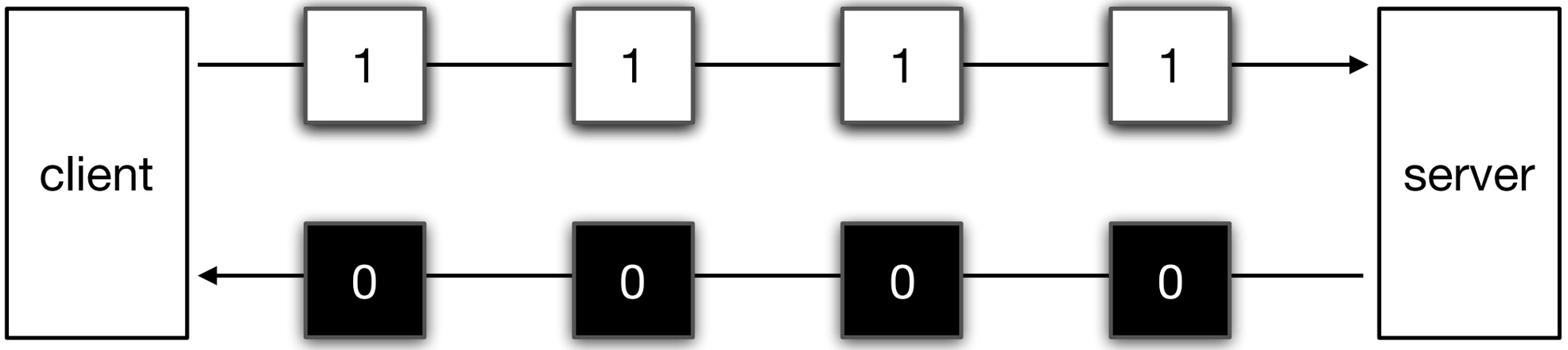
How does it work?



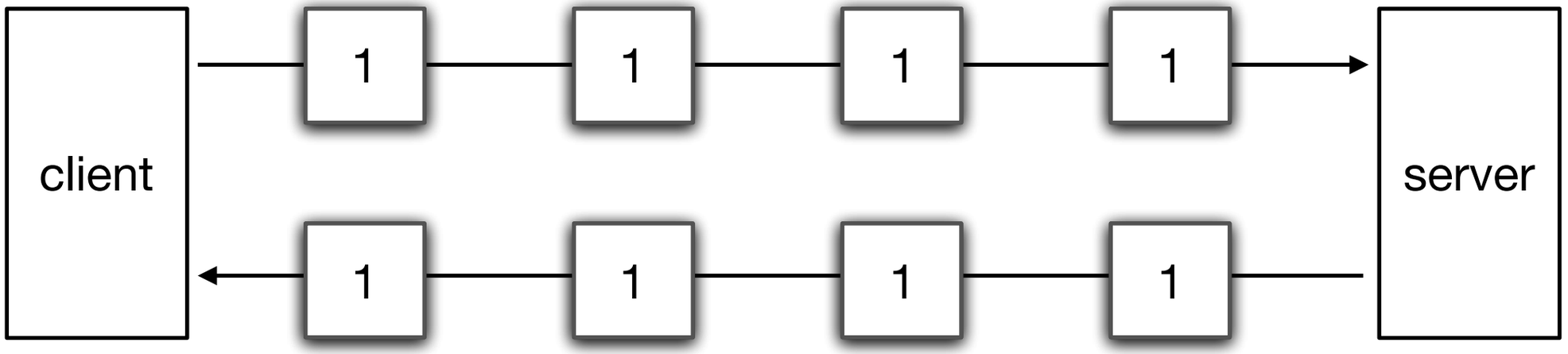
How does it work?



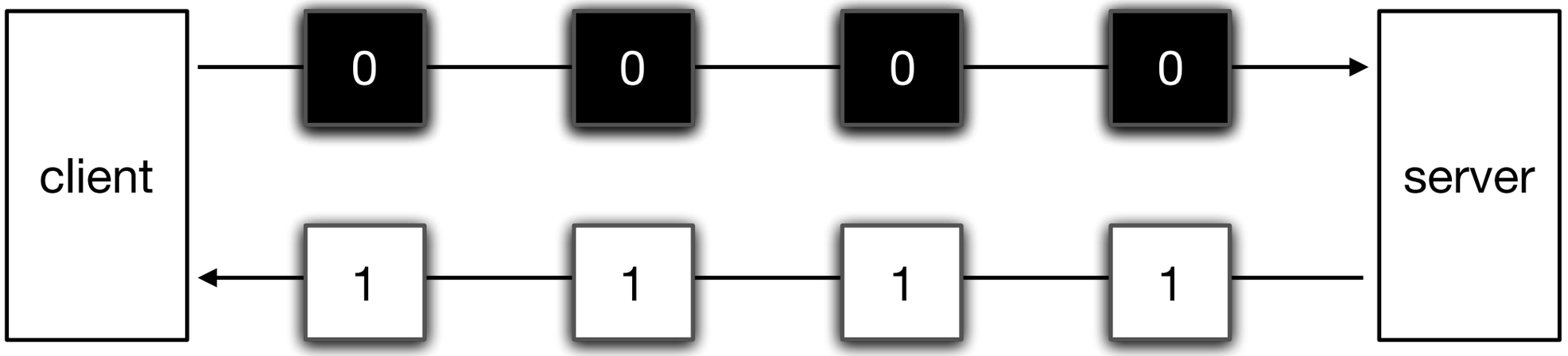
How does it work?



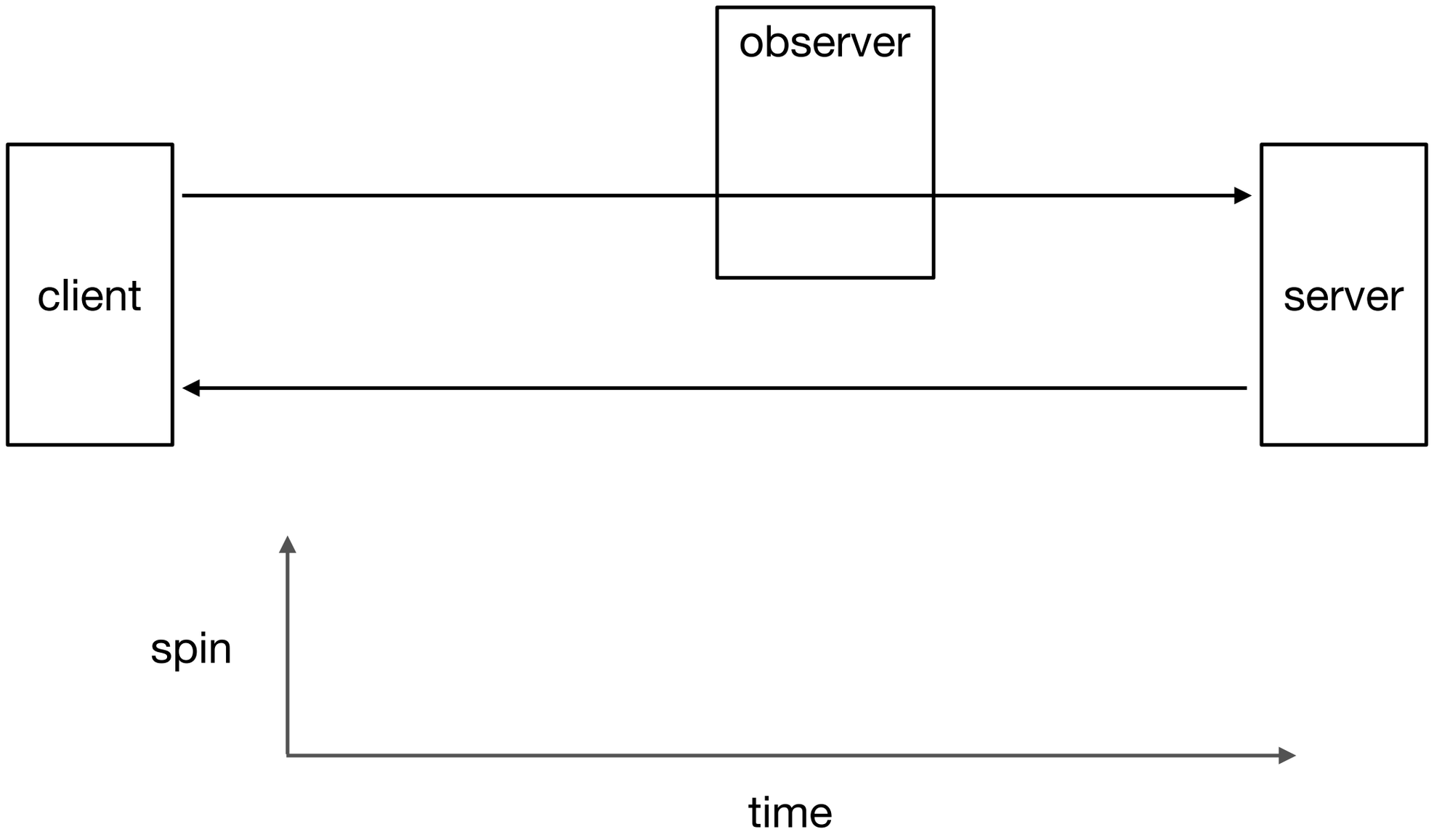
How does it work?



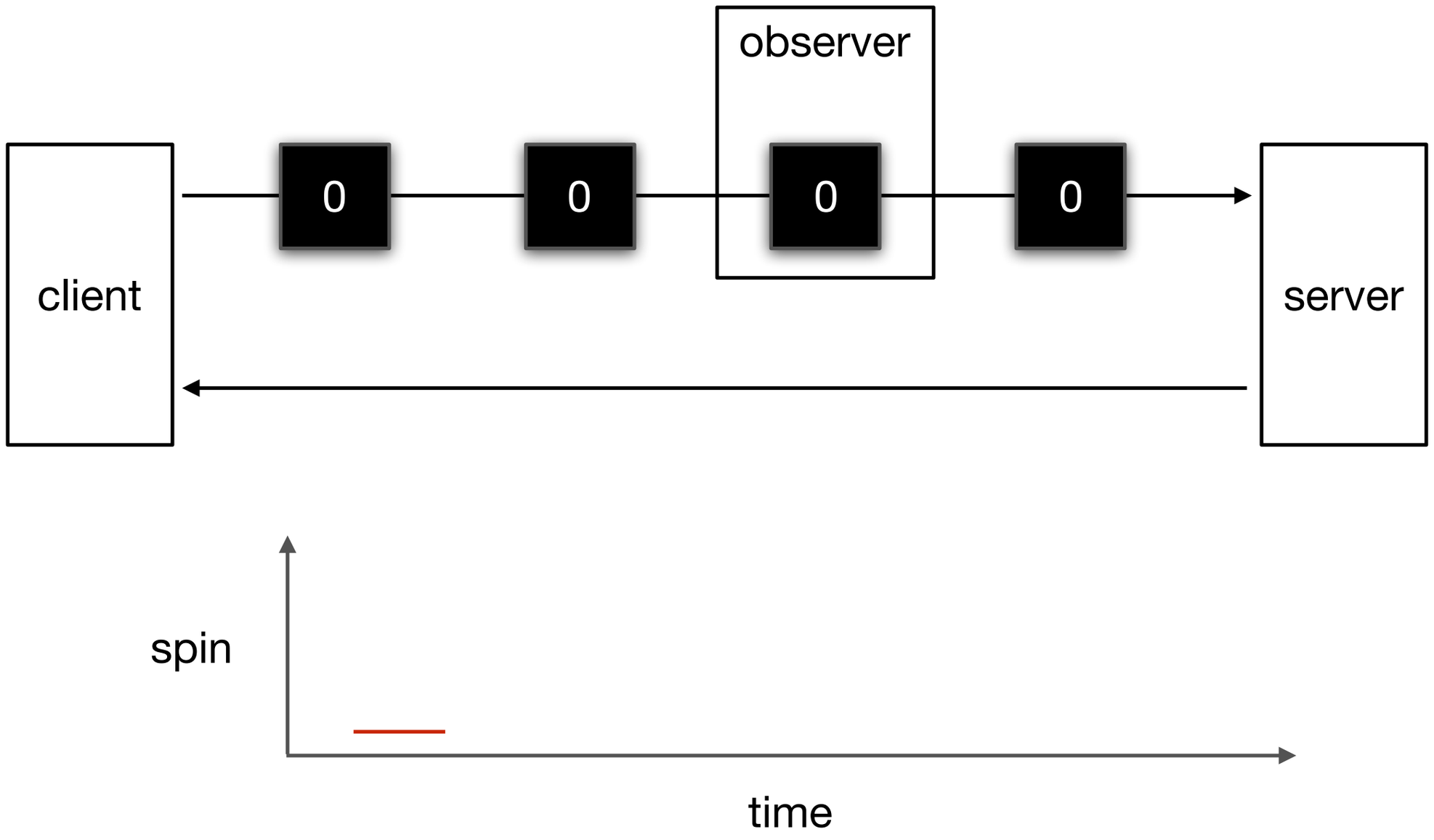
How does it work?



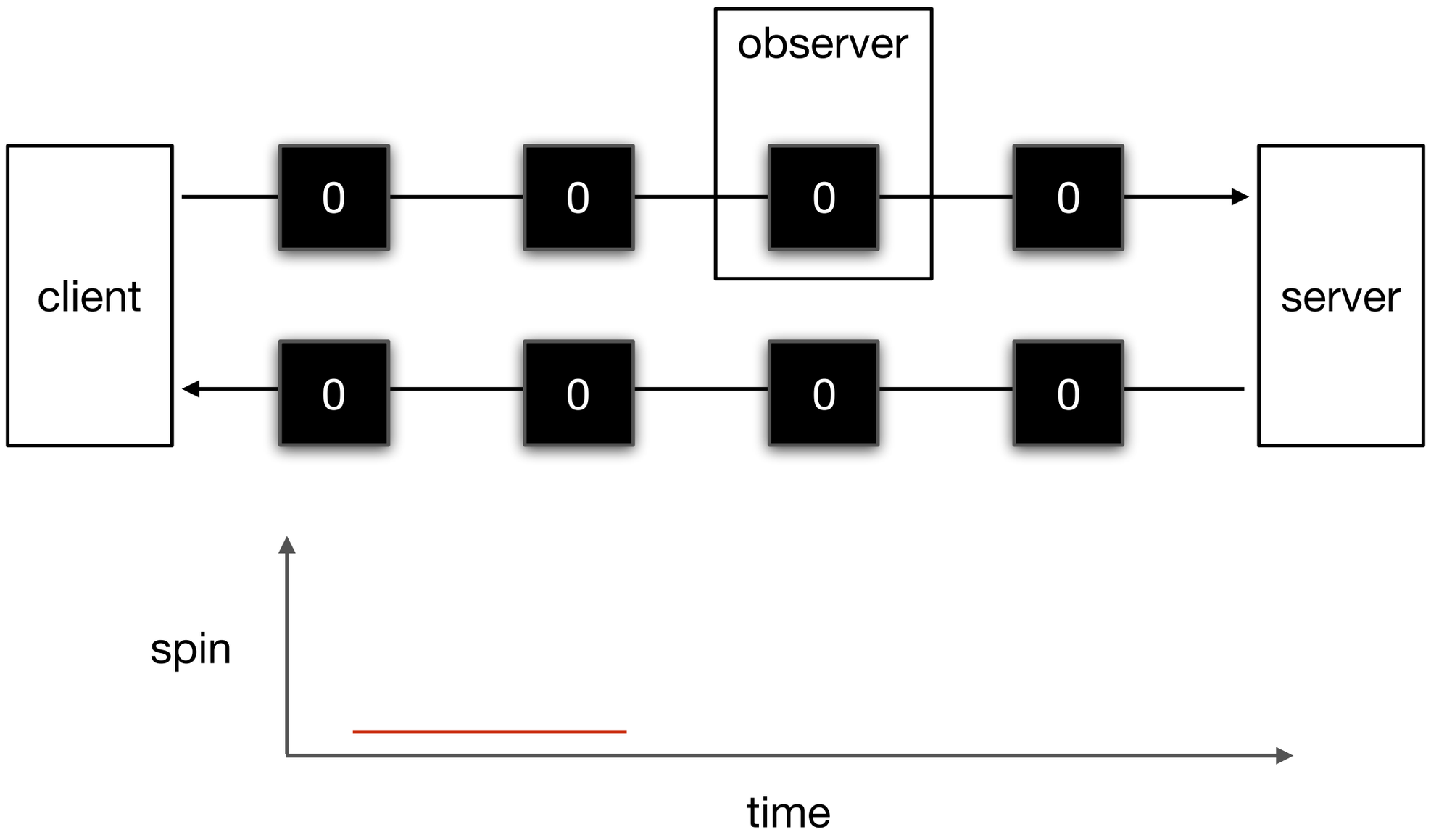
Unidirectional one-point measurement



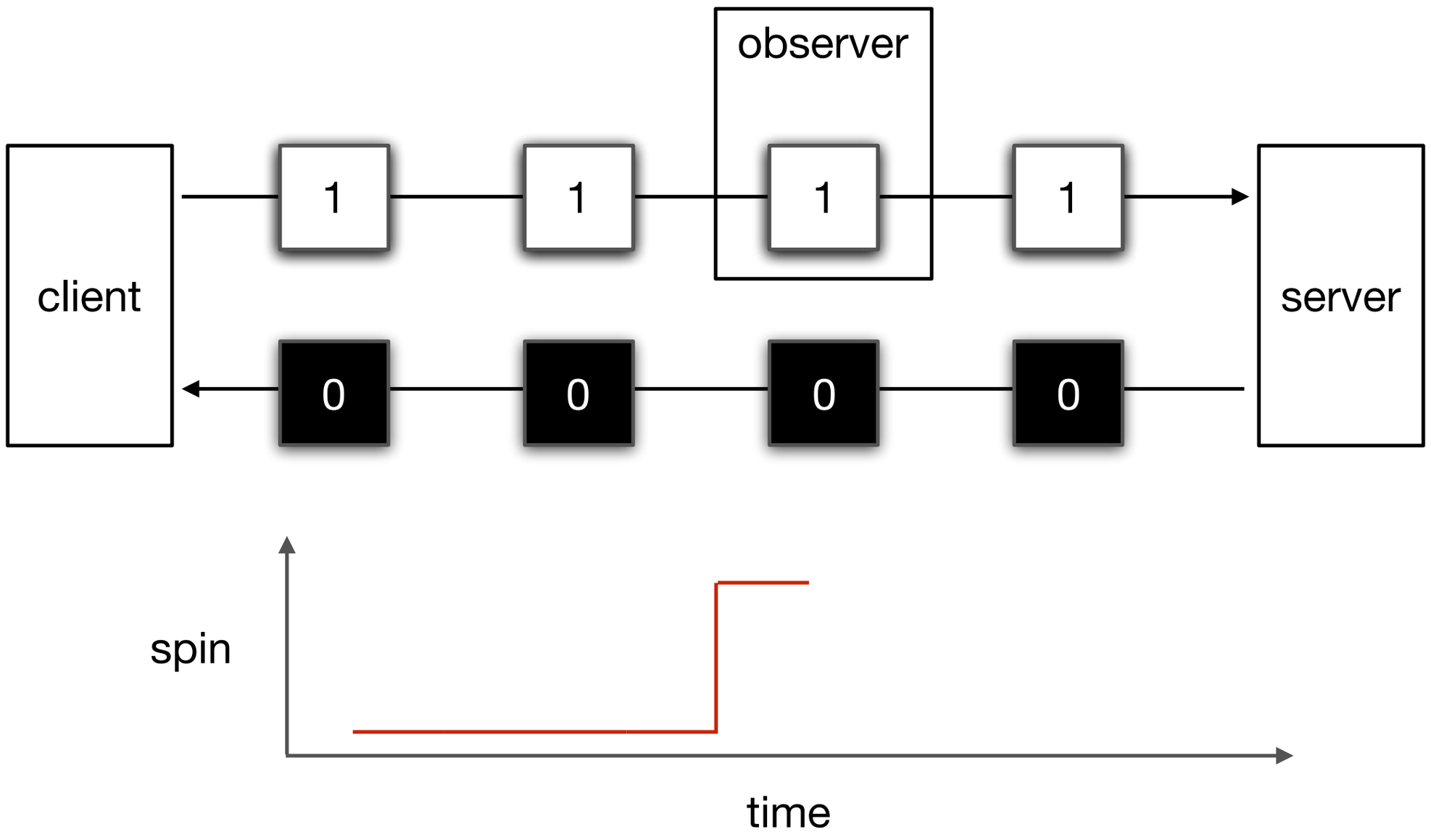
Unidirectional one-point measurement



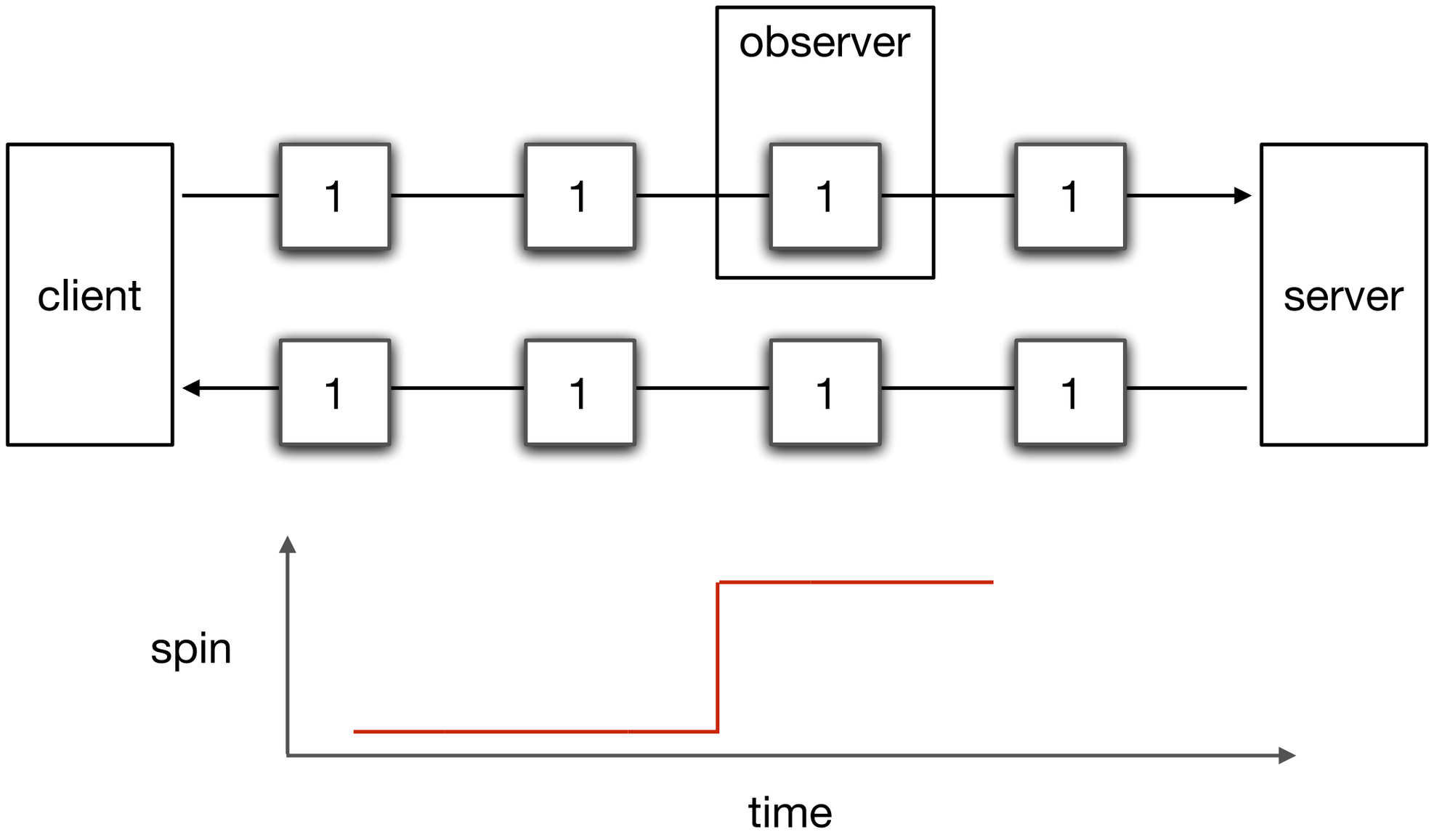
Unidirectional one-point measurement



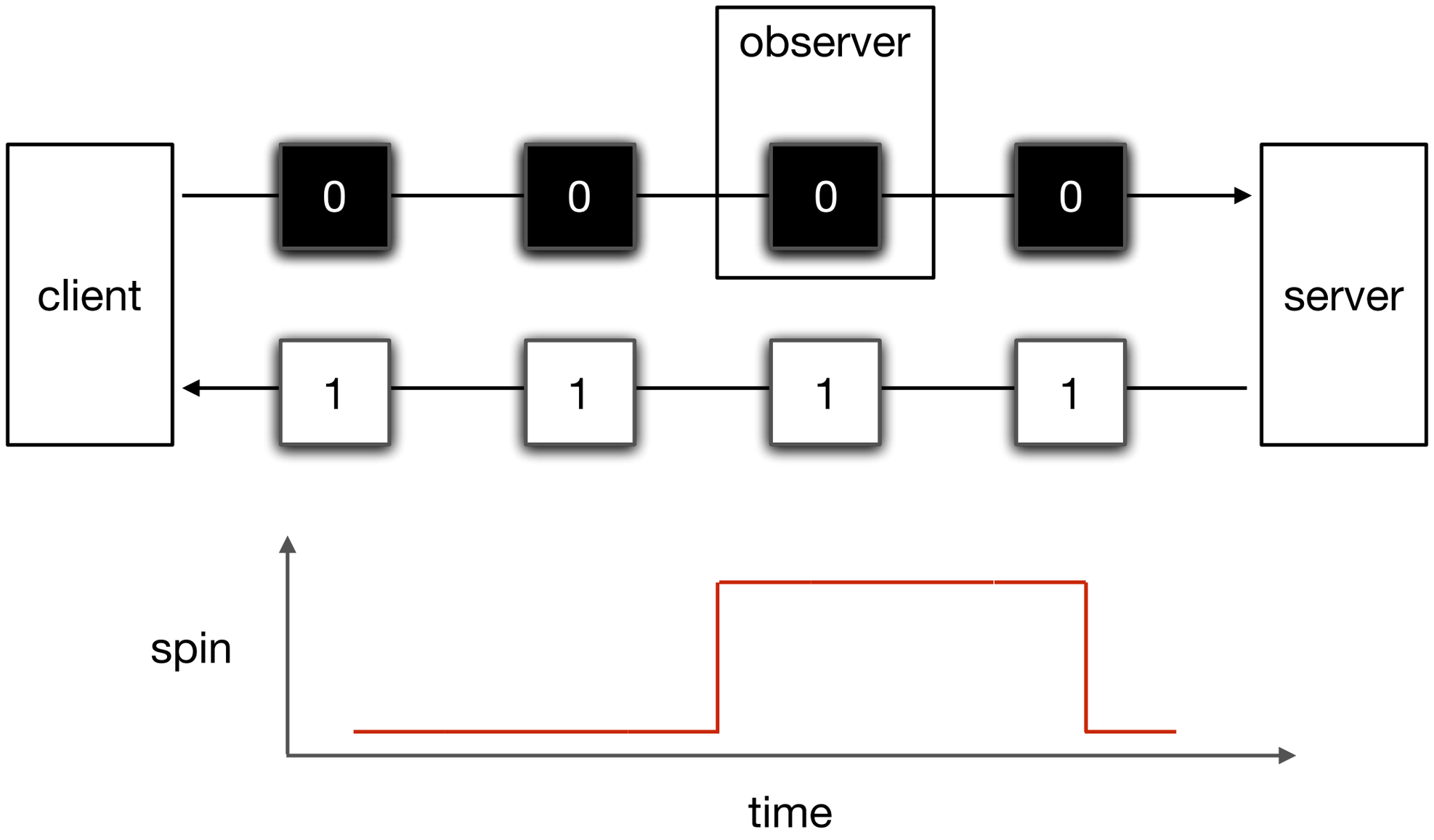
Unidirectional one-point measurement



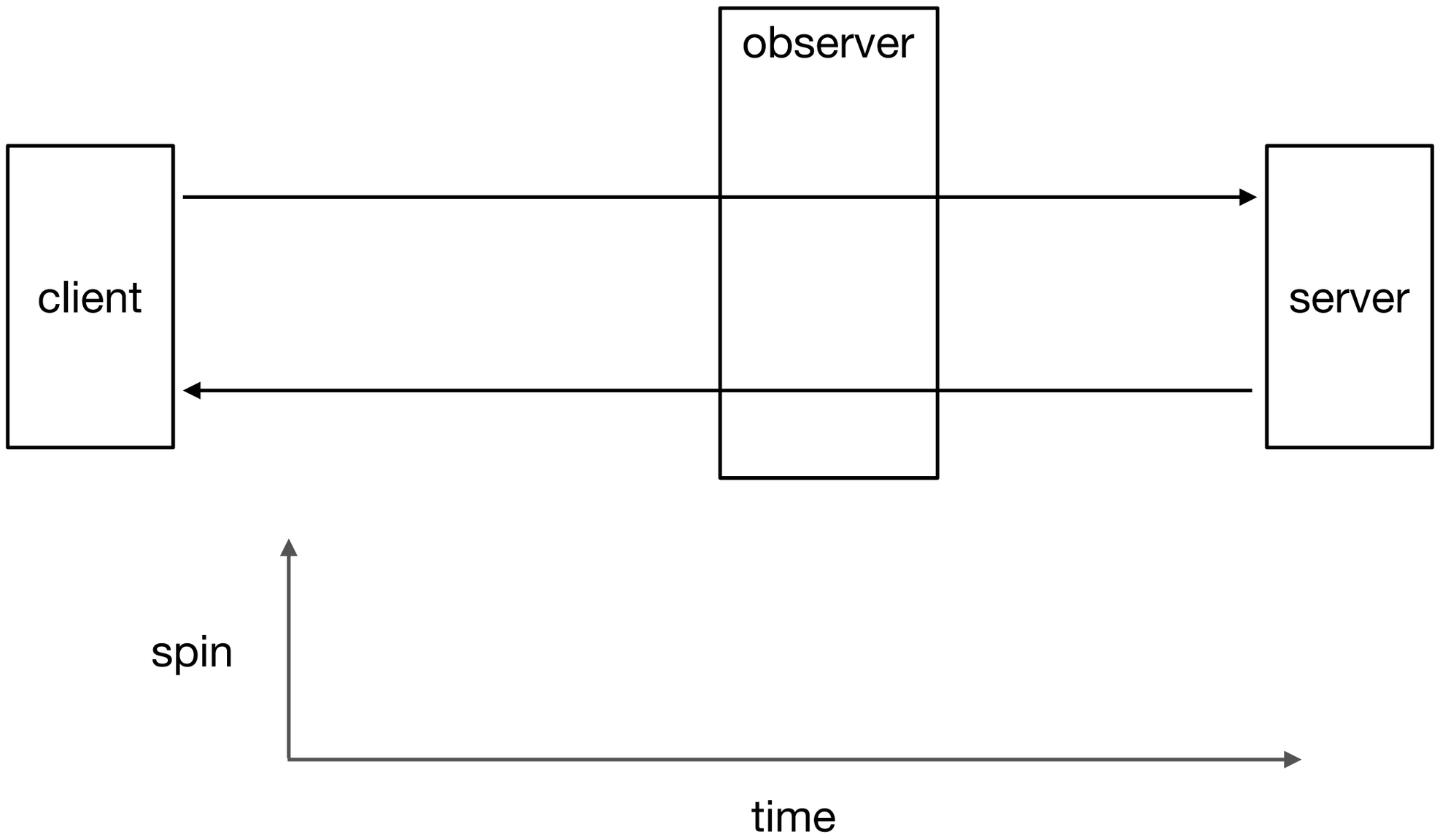
Unidirectional one-point measurement



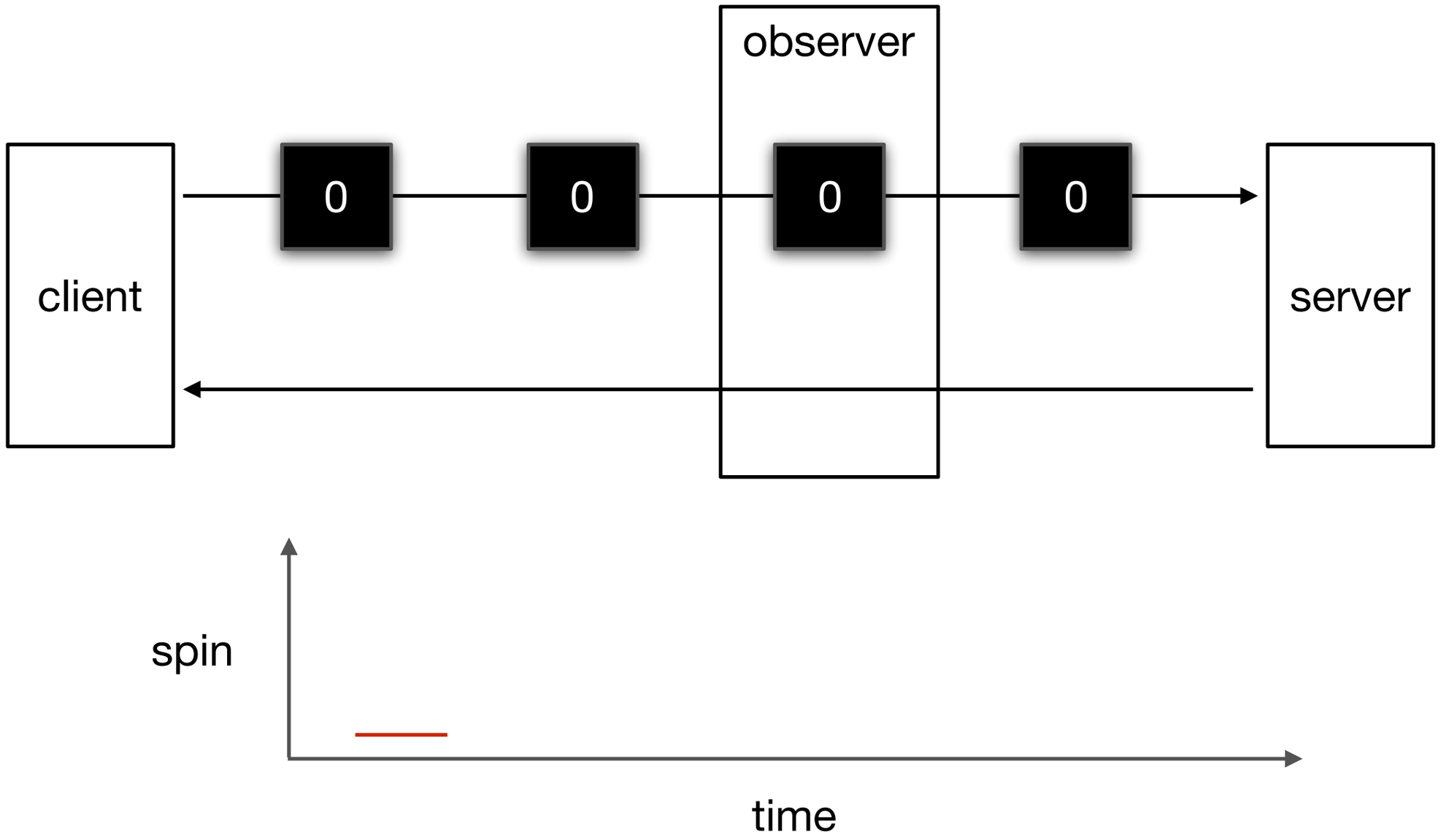
Unidirectional one-point measurement



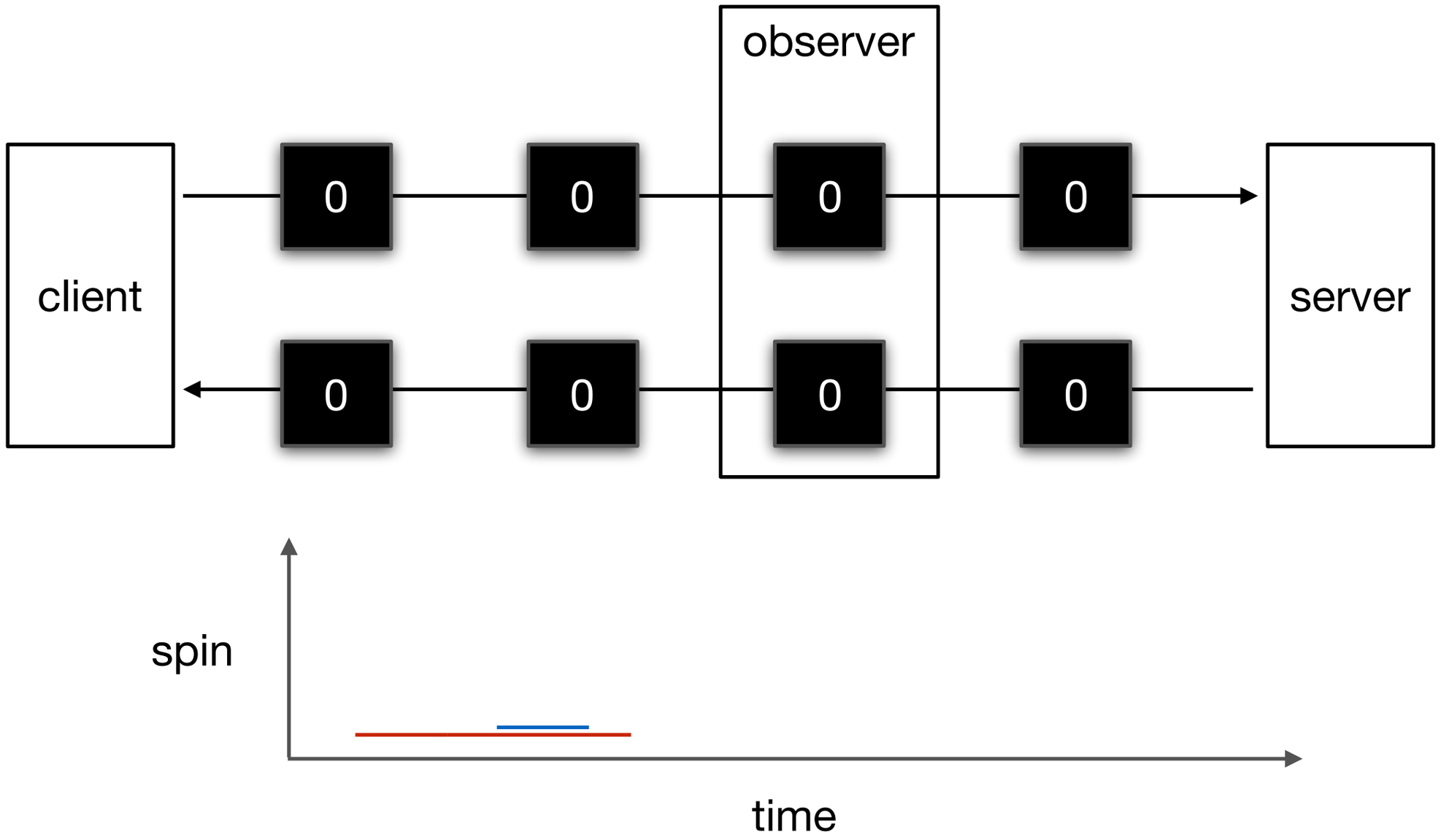
Bidirectional one-point measurement



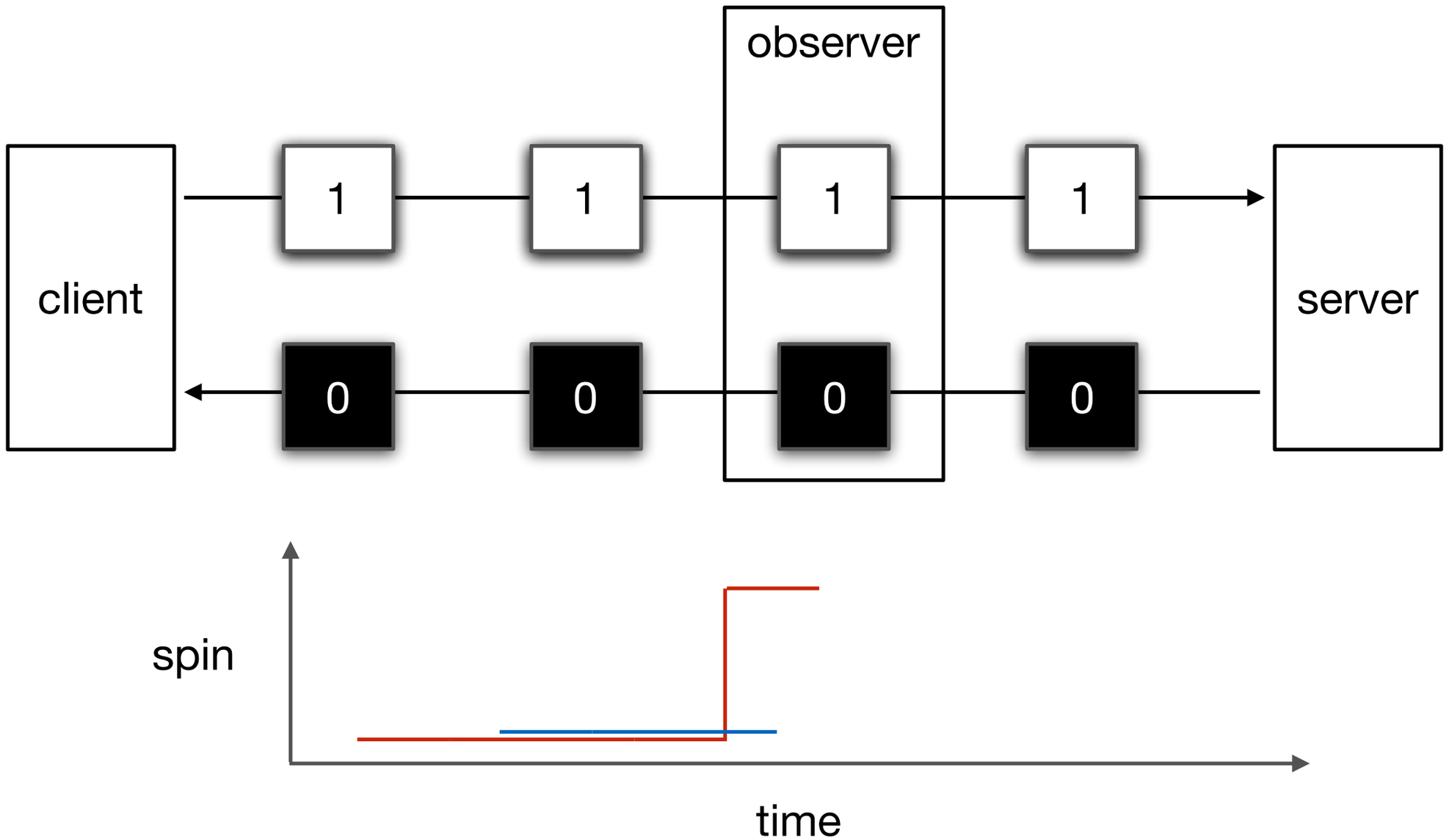
Bidirectional one-point measurement



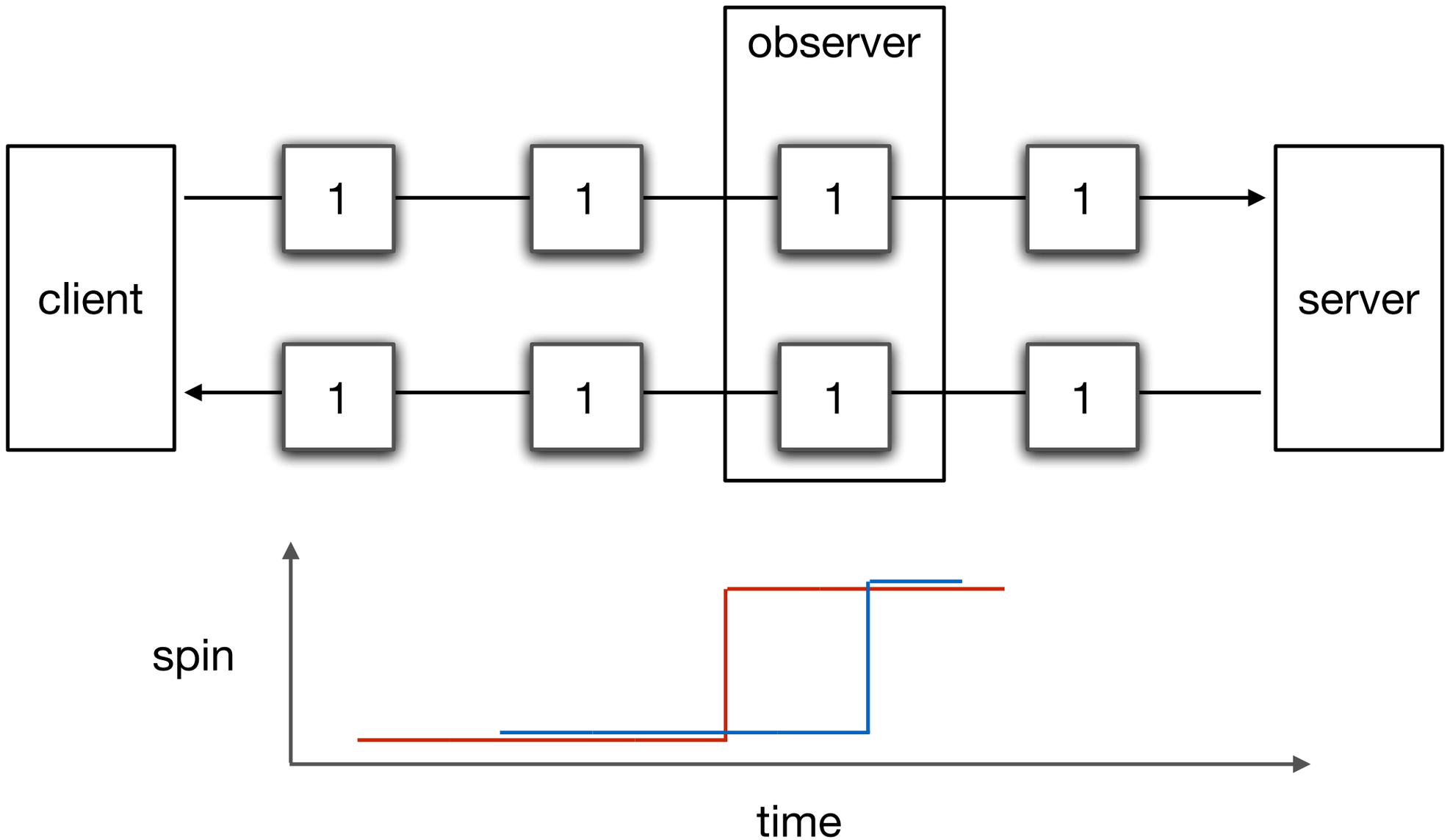
Bidirectional one-point measurement



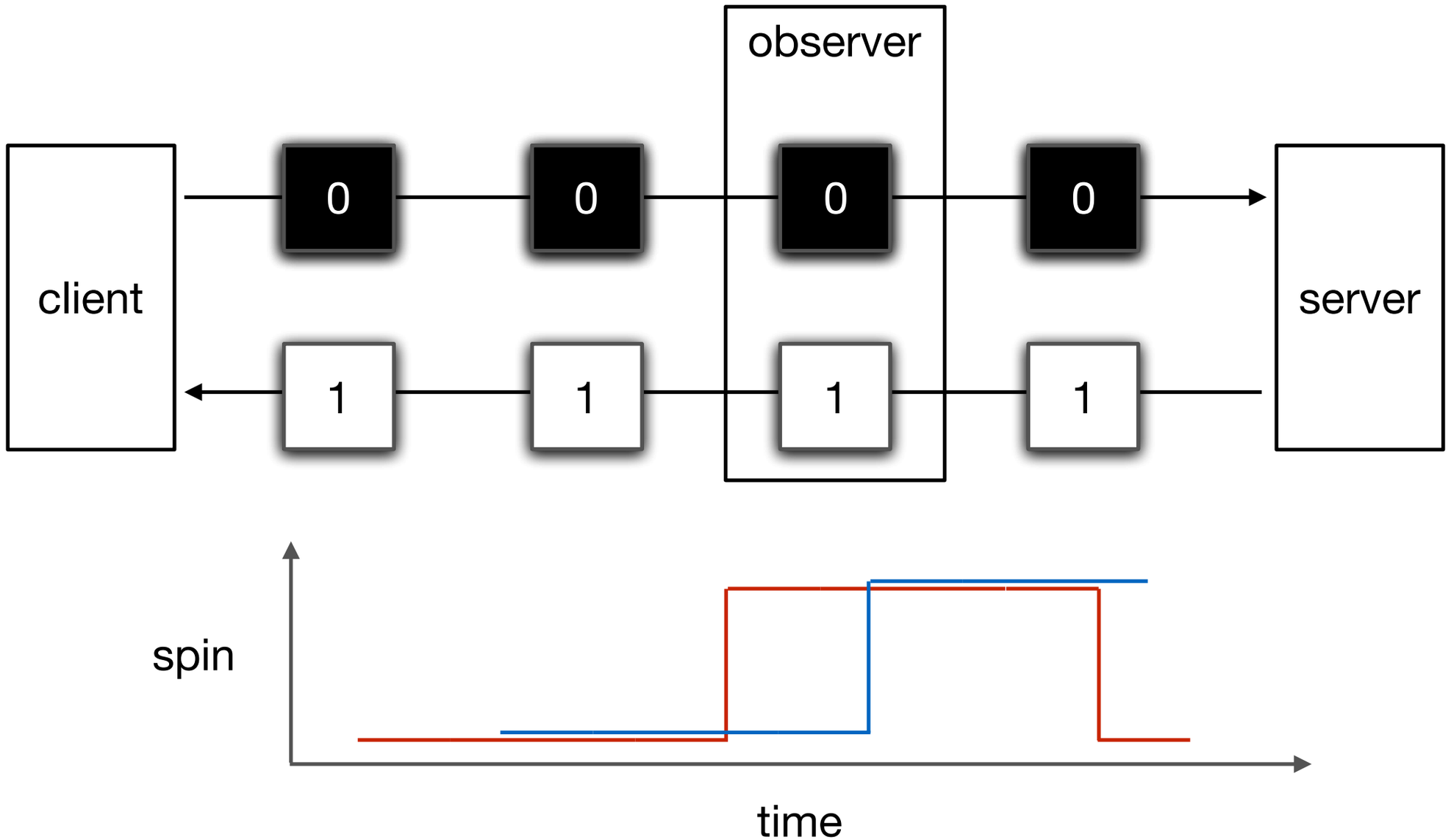
Bidirectional one-point measurement



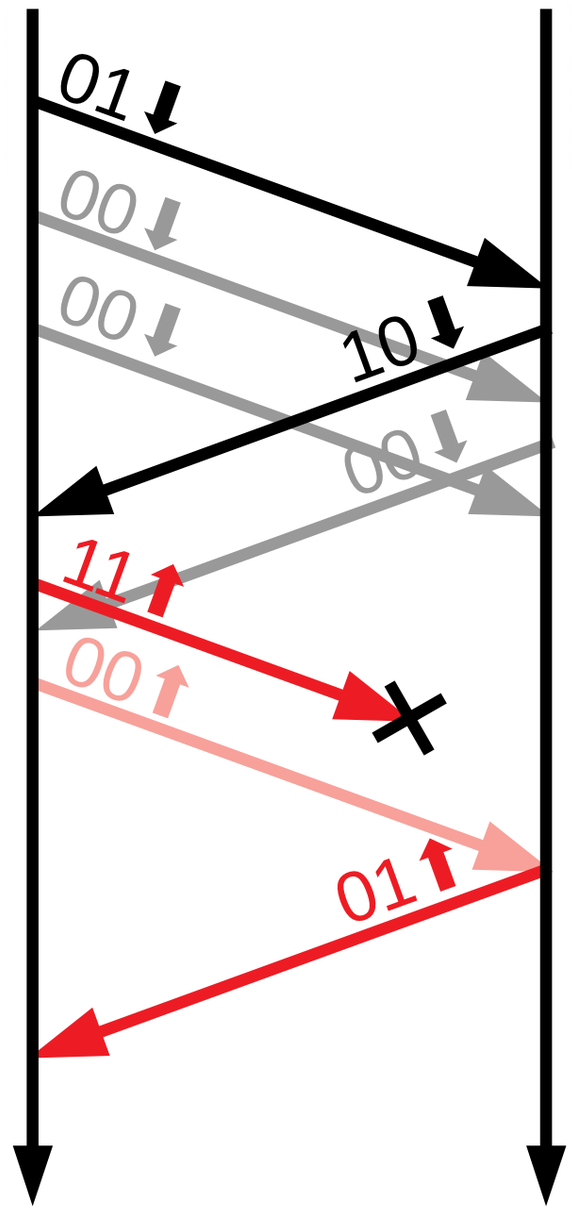
Bidirectional one-point measurement



Bidirectional one-point measurement



Dealing with Loss and Reordering: The Valid Edge Counter



- Bursty traffic can lead to wild overestimates of RTT: adds delay between bursts to actual measured RTT.
  - A damping filter can reduce overestimate samples
- Addition of a two-bit *valid edge counter* eliminates overestimation as well as fixing issues with packet loss and reordering:
  - On non-edge, delayed edge, edge on reordered packet: valid ← 00
  - On all other edges: valid ← last received valid + 1
  - Produces a 11 signal ("good edge") 1.5RTT after last reorder/ delay, requires both sides to be reordering/delay-free, resets after an edge is lost.
- Rejects invalid samples due to bursty traffic, deals with reordering as well as two-bit spin, and adds tolerance to heavy burst losses, without PN visibility

Summary

- New protocol suite for web traffic is optimized for latency and provides increased security
  - *Mutlistreaming* to avoid Head-of-line Blocking in **HTTP2** and **QUIC**
  - *0-RTT support* and *Perfect Forward Secrecy* in **TLS1.3**
  - Encryption integrated in the transport layer with **QUIC**

- Limited information available in the network visible **wire image**
  - *Explicit measurability* to replace implicit assumptions about the internal protocol machinery