The Latency Spin Bit

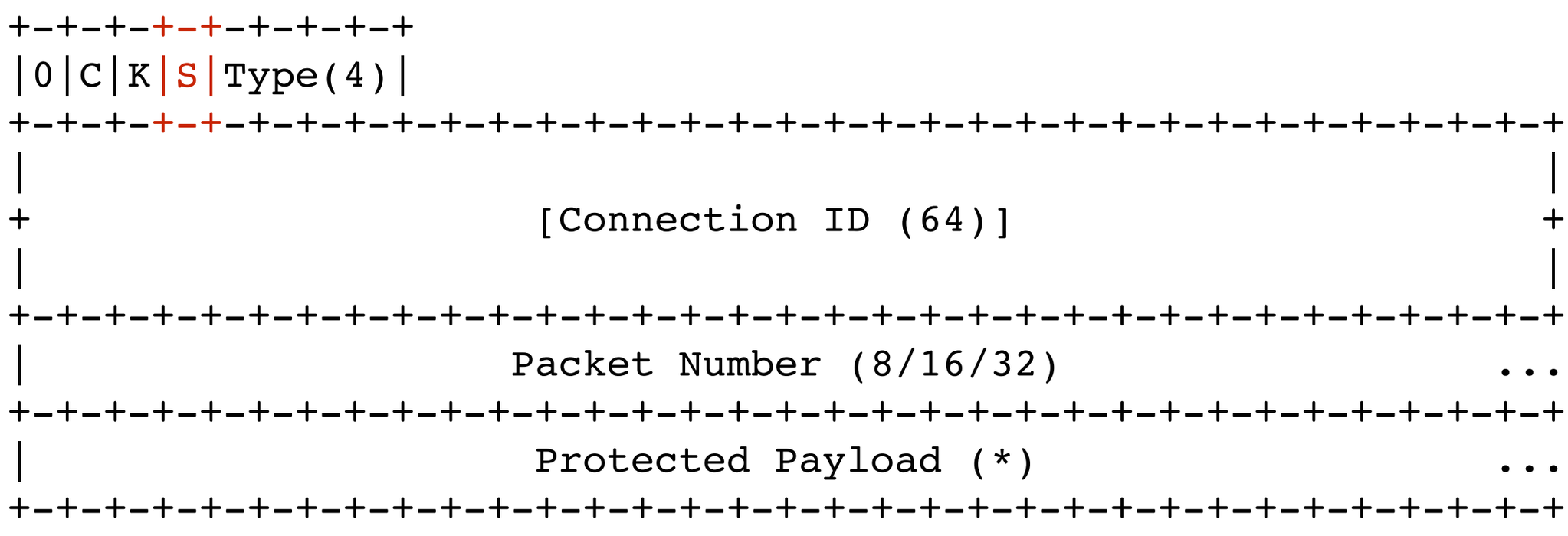
draft-trammell-quic-spin-01

Brian Trammell — ETH Zürich
(with Piet De Vaere)

IETF 101 London

What is it?

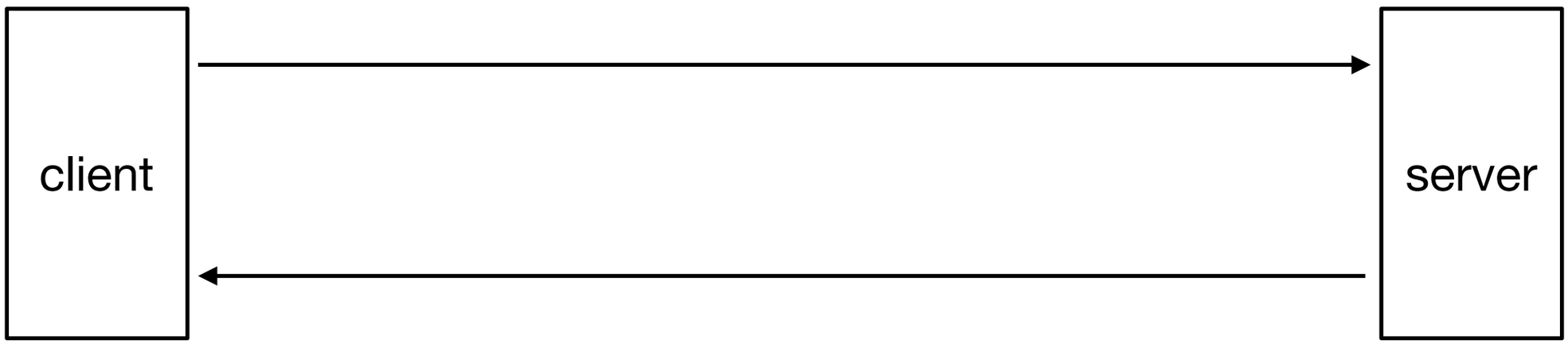
- Proposal (#1046): take a bit from QUIC short header type field and make it spin
- Server sets last spin it saw on each packet it sends
- Client sets ~(last spin it saw) on each packet it sends
- Creates a square-wave with period == RTT (when sender not app-limited)



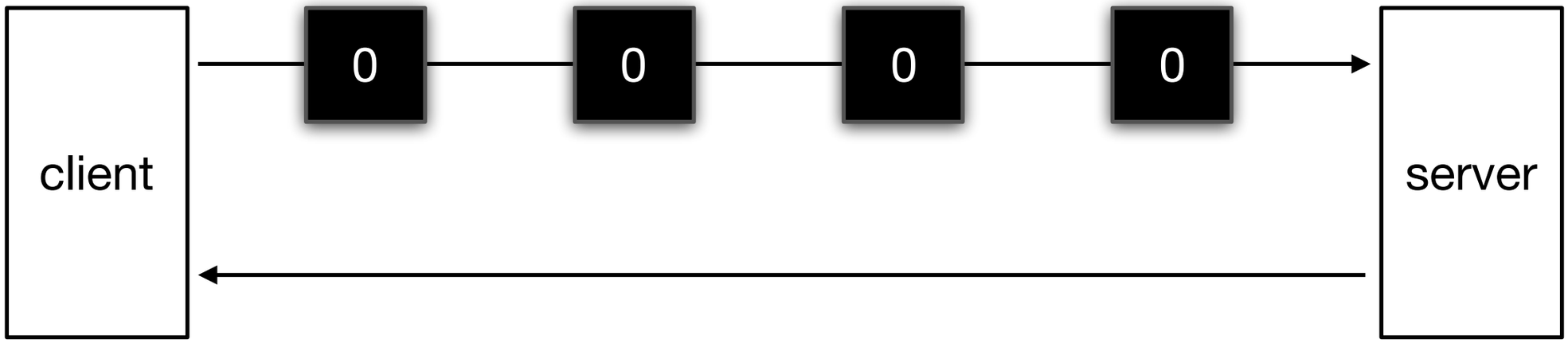
Why?

- Explicit signal for passive measurement of per-flow RTT
- ***Reduce loss of visibility*** of metrics with respect to TCP:
  - Replaces SEQ/ACK or TSval/TSecr calculation in TCP
  - Superior to QUIC handshake RTT: multiple samples per flow, no additional handshake-linked delay
- Use cases enumerated in draft-trammell-quic-spin-01:
  - Interdomain and intradomain troubleshooting
  - Home network troubleshooting
  - Bufferbloat mitigation for mobile networks
  - Internet measurement research

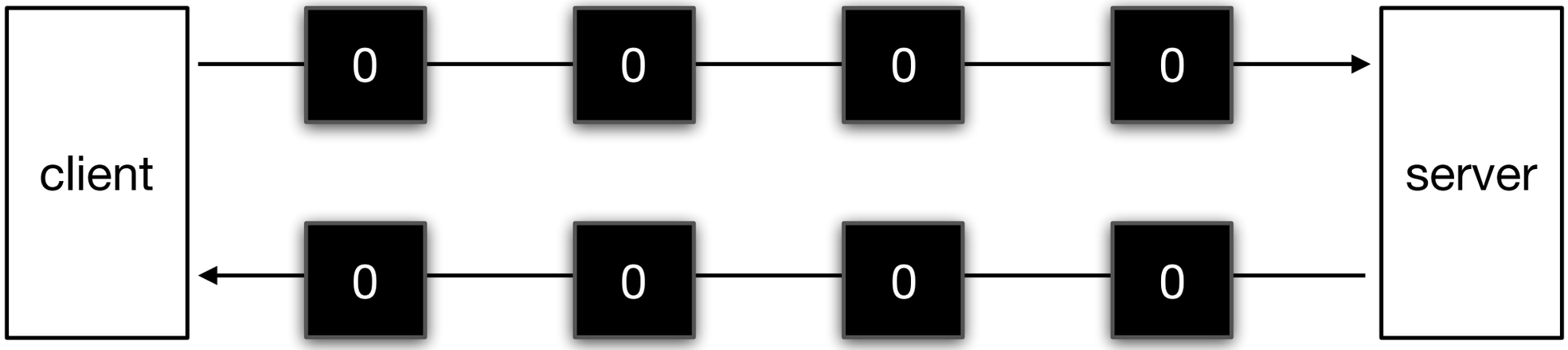
How does it work?



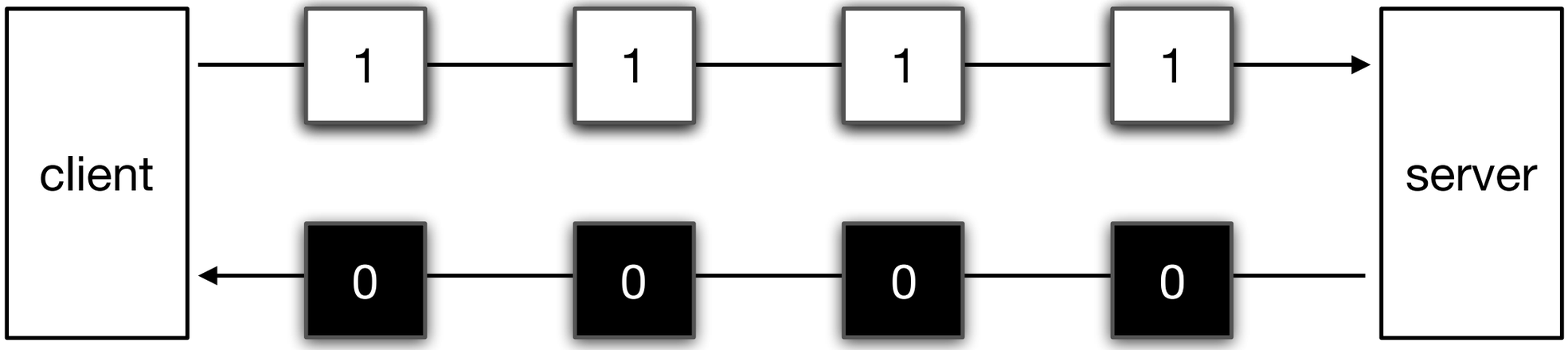
How does it work?



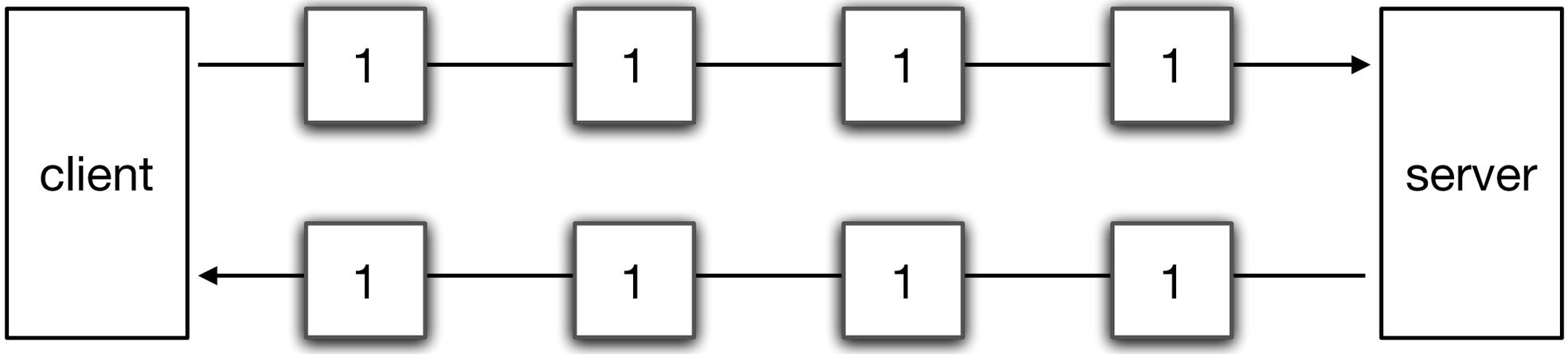
How does it work?



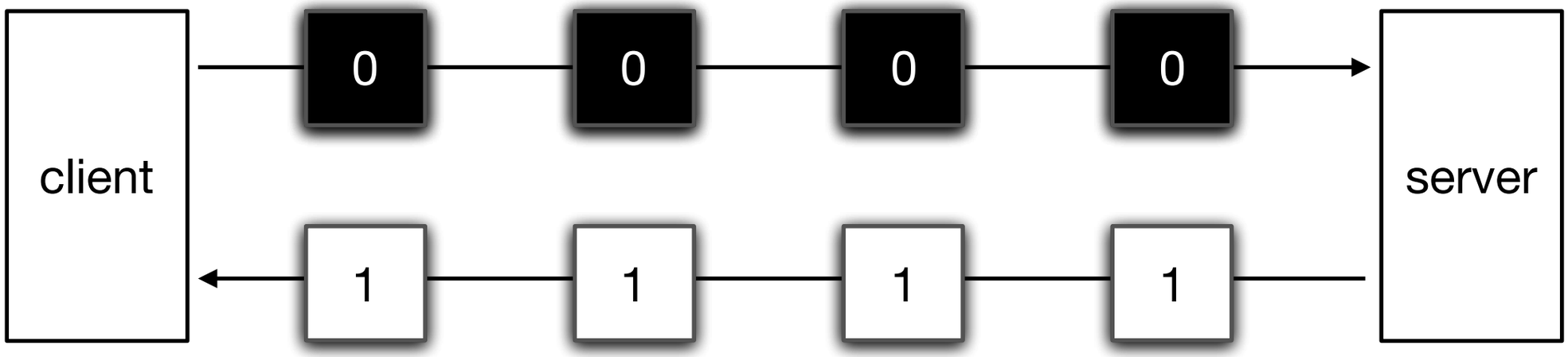
How does it work?



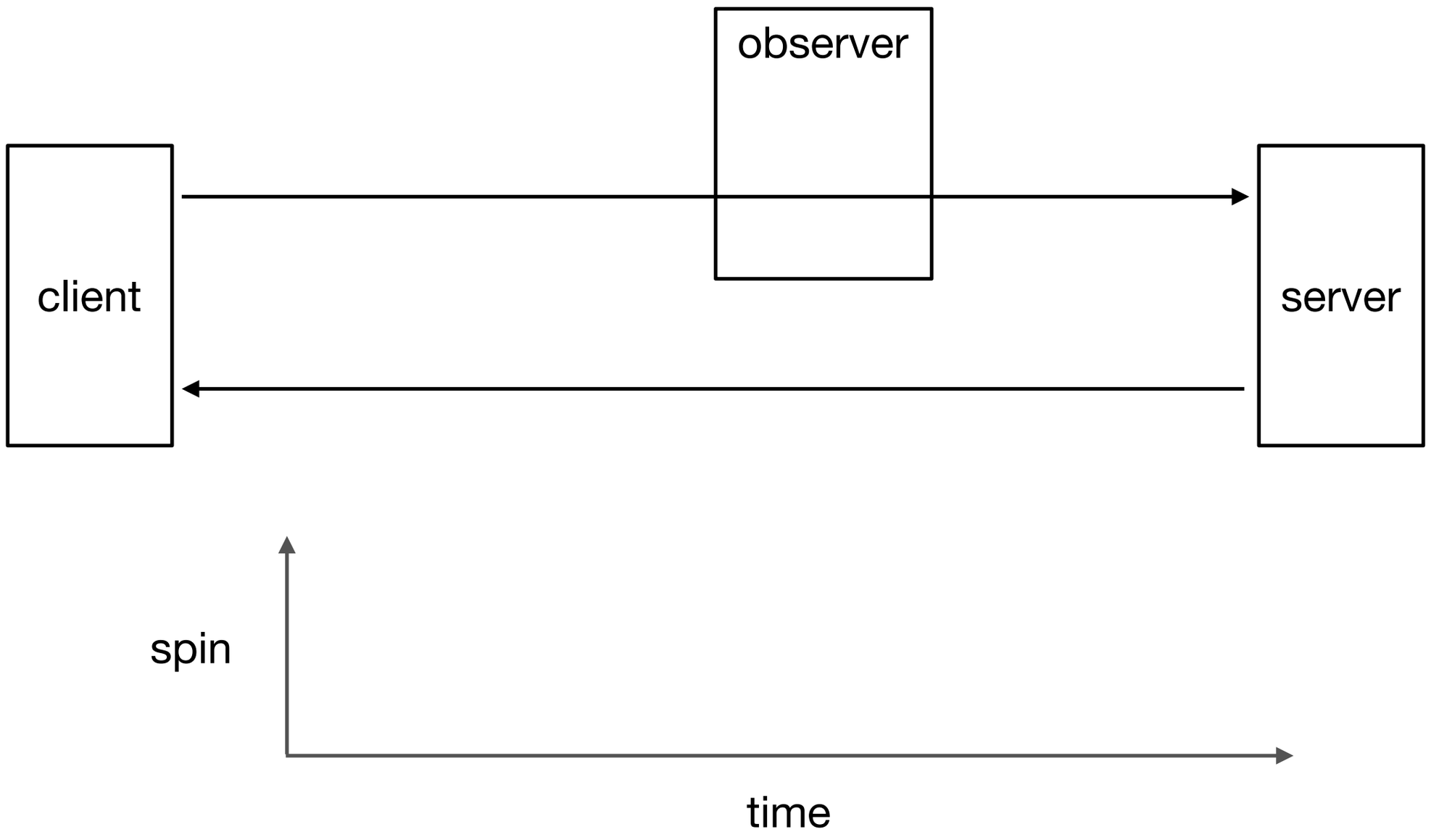
How does it work?



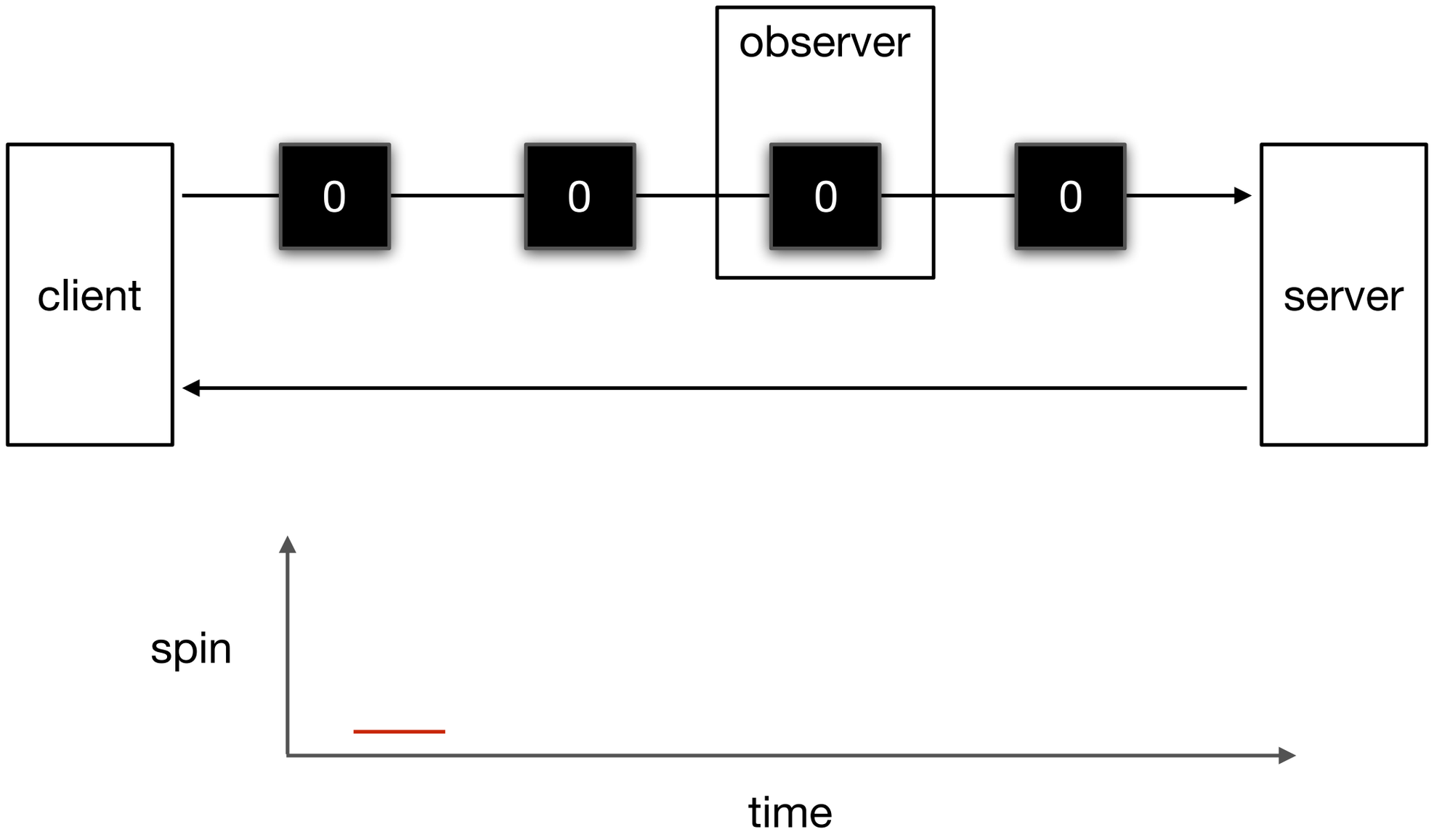
How does it work?



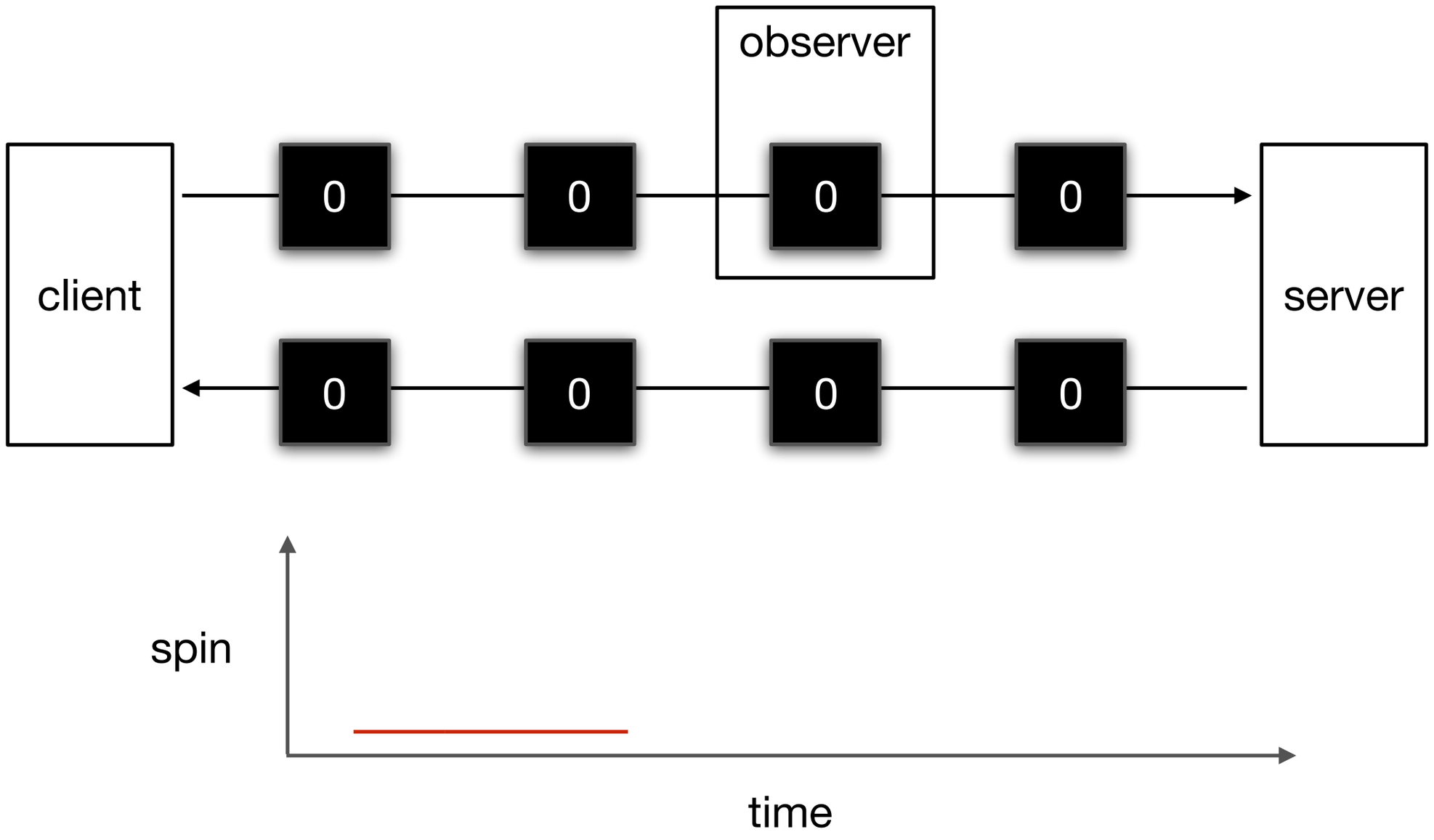
Unidirectional one-point measurement



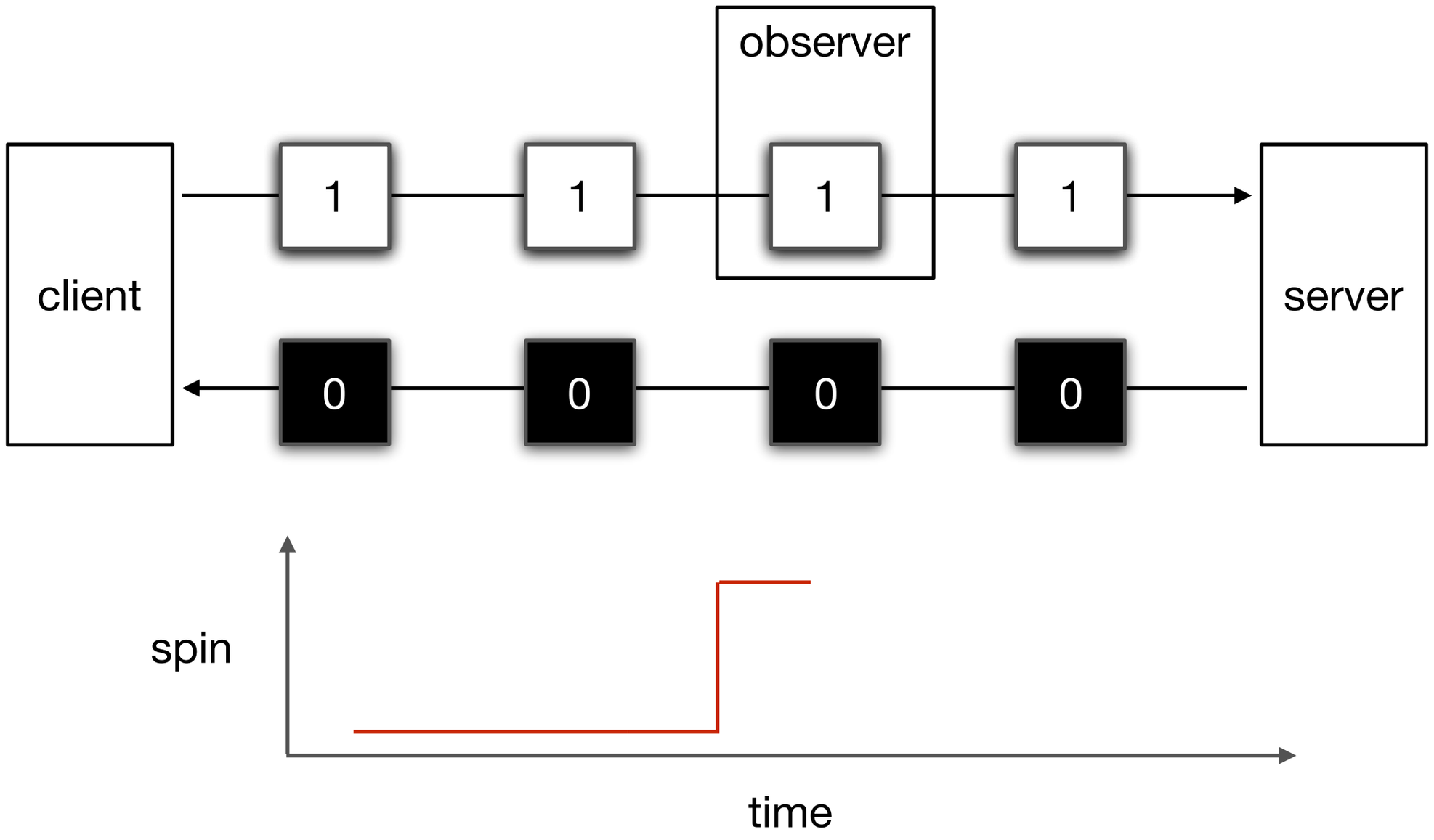
Unidirectional one-point measurement



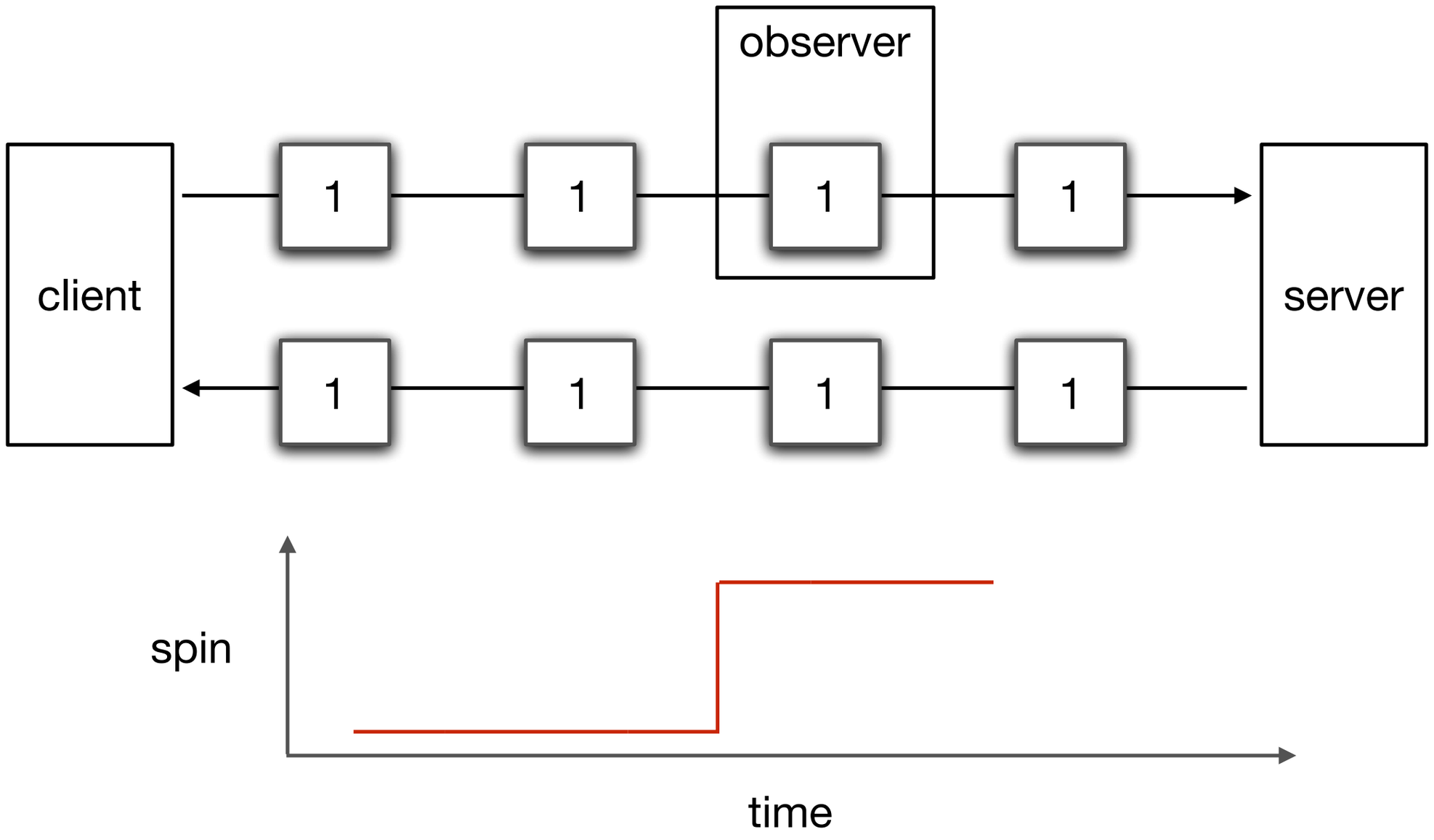
Unidirectional one-point measurement



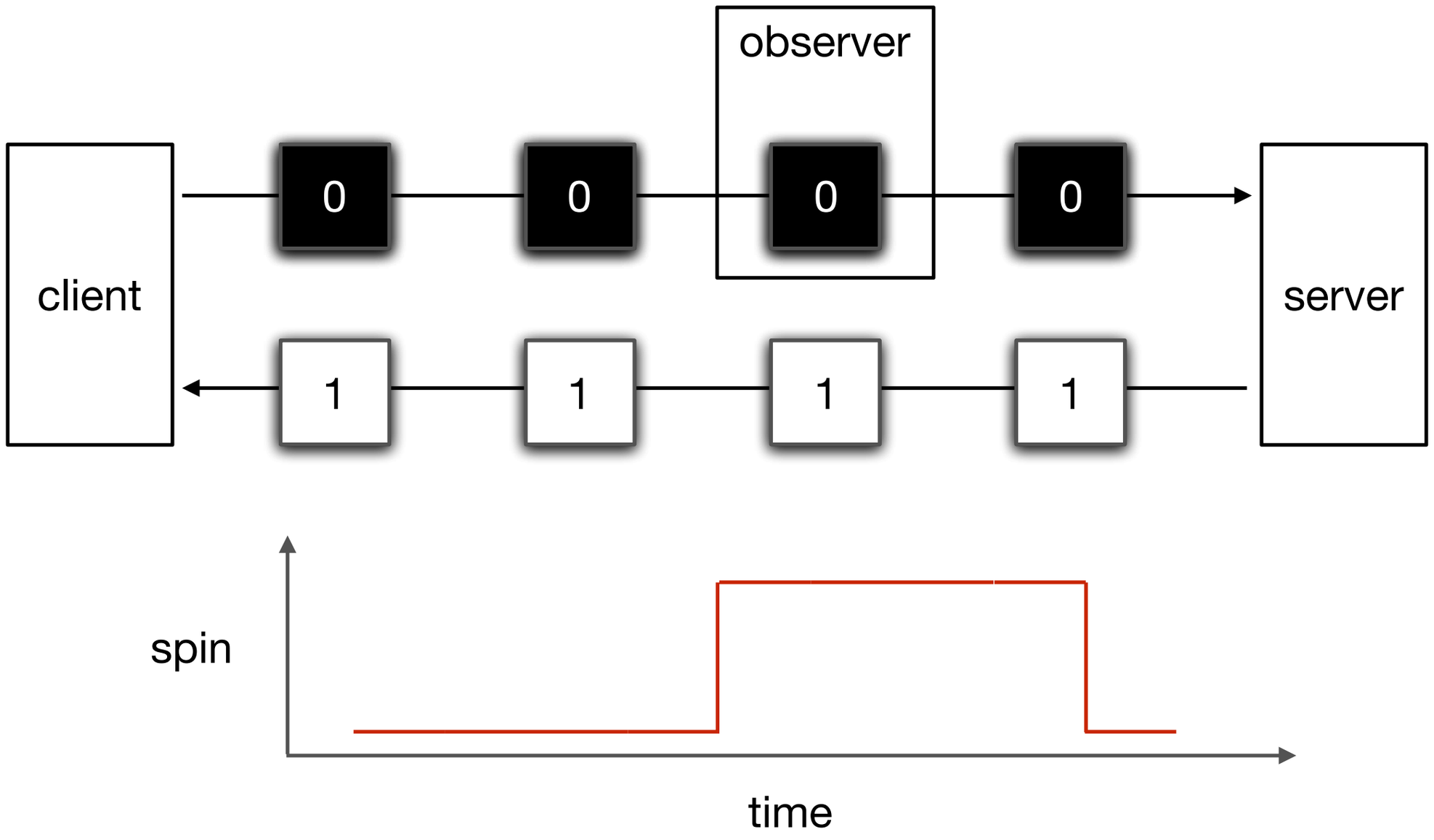
Unidirectional one-point measurement



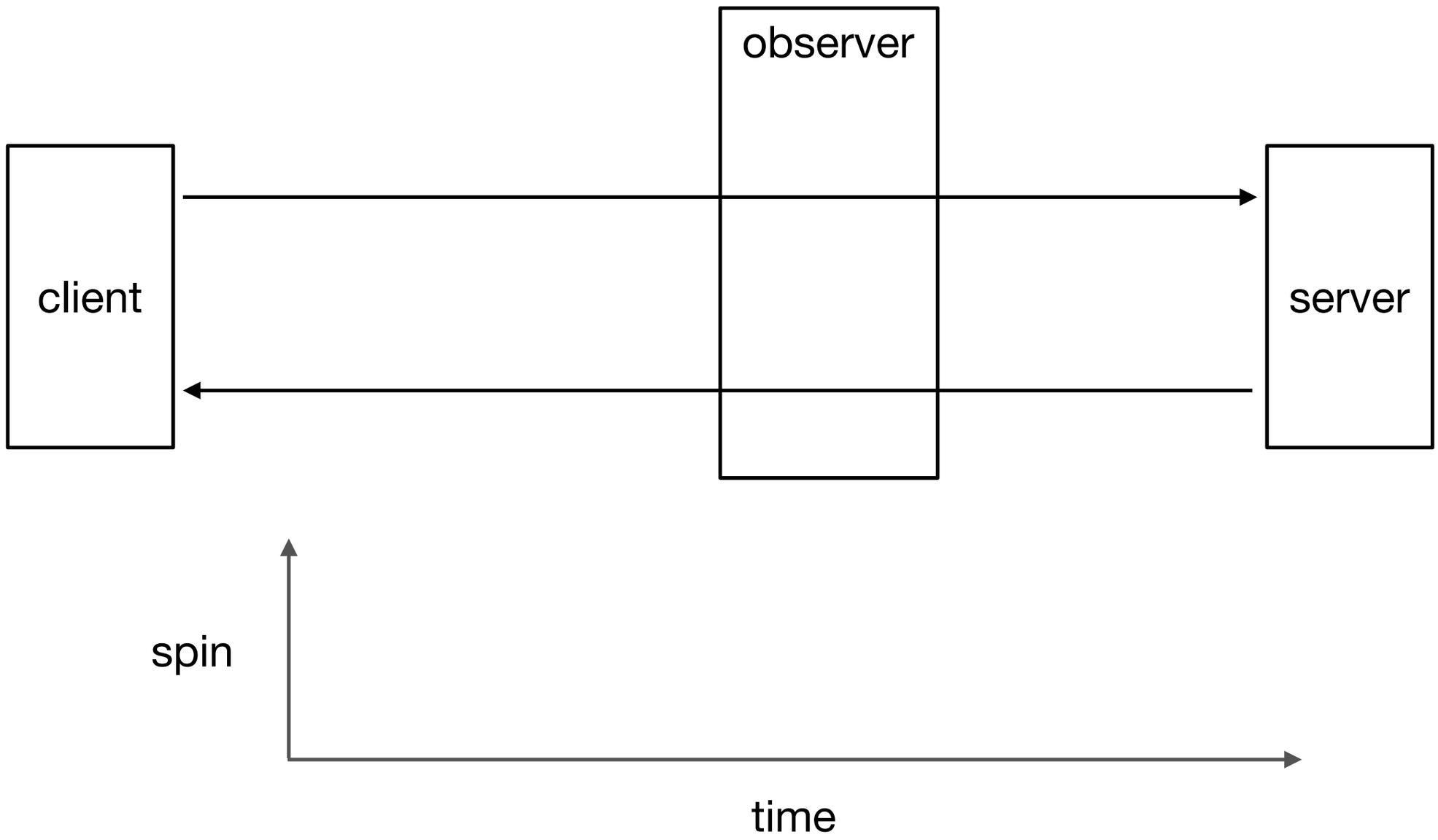
Unidirectional one-point measurement



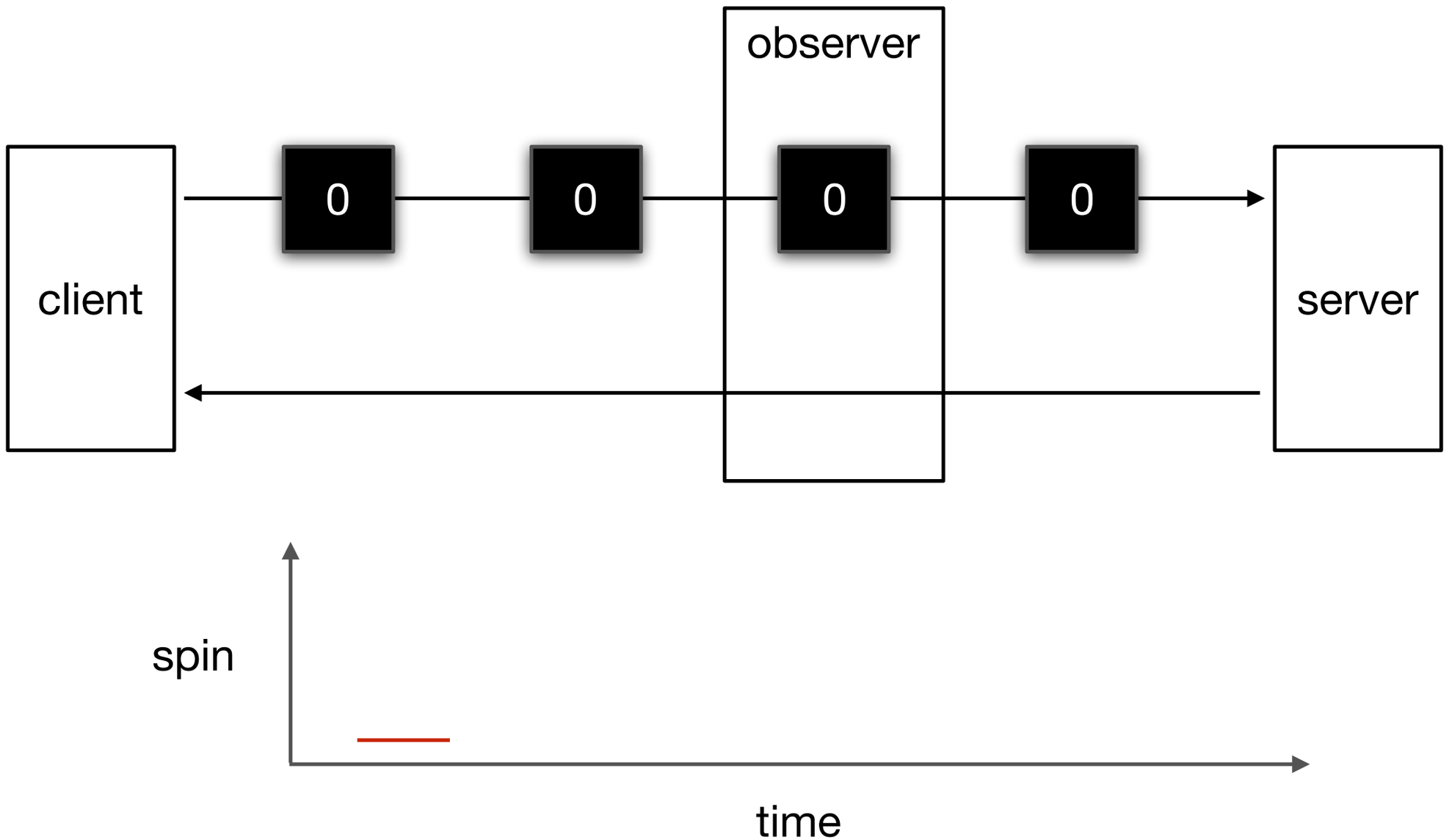
Unidirectional one-point measurement



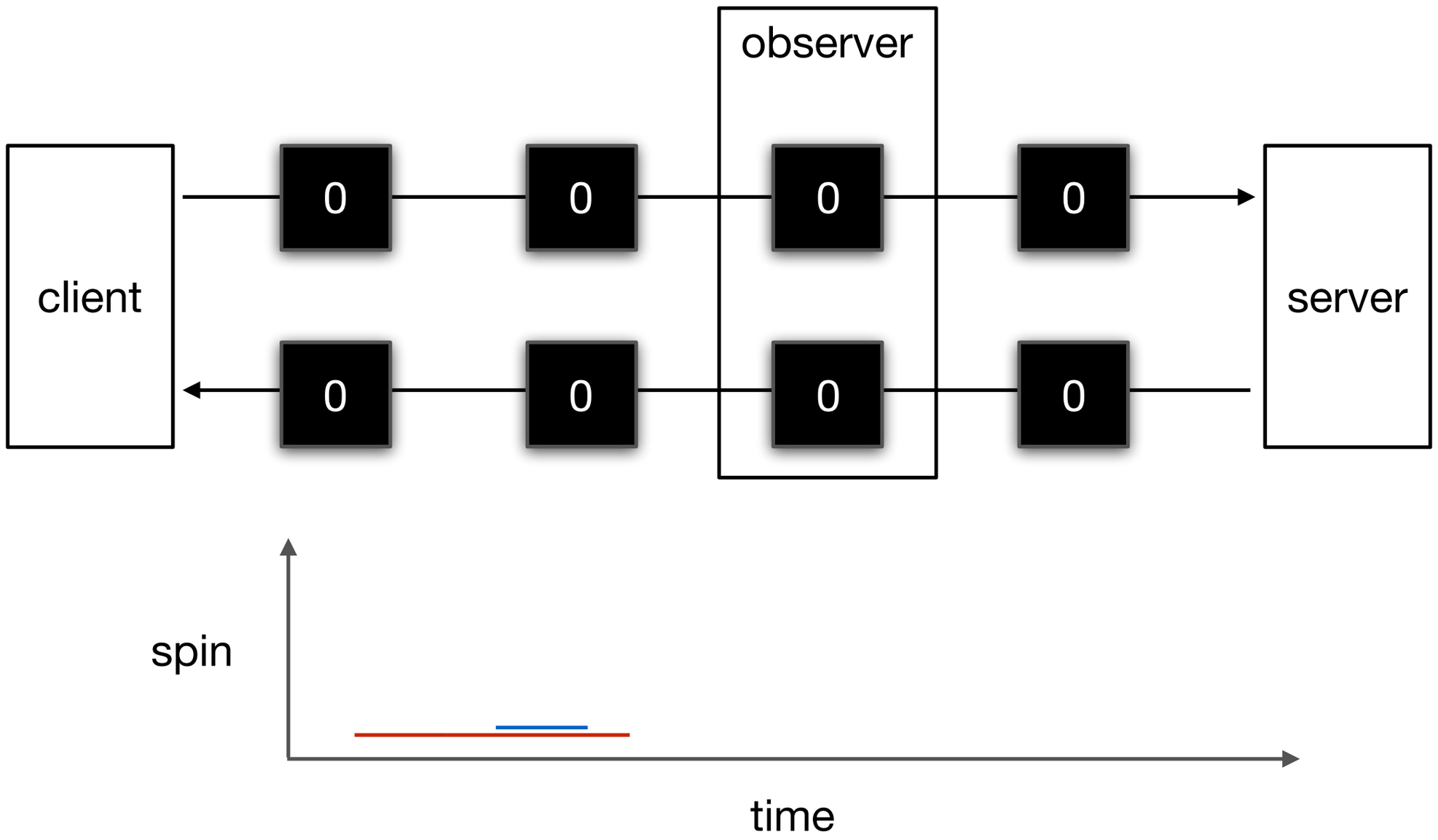
Bidirectional one-point measurement



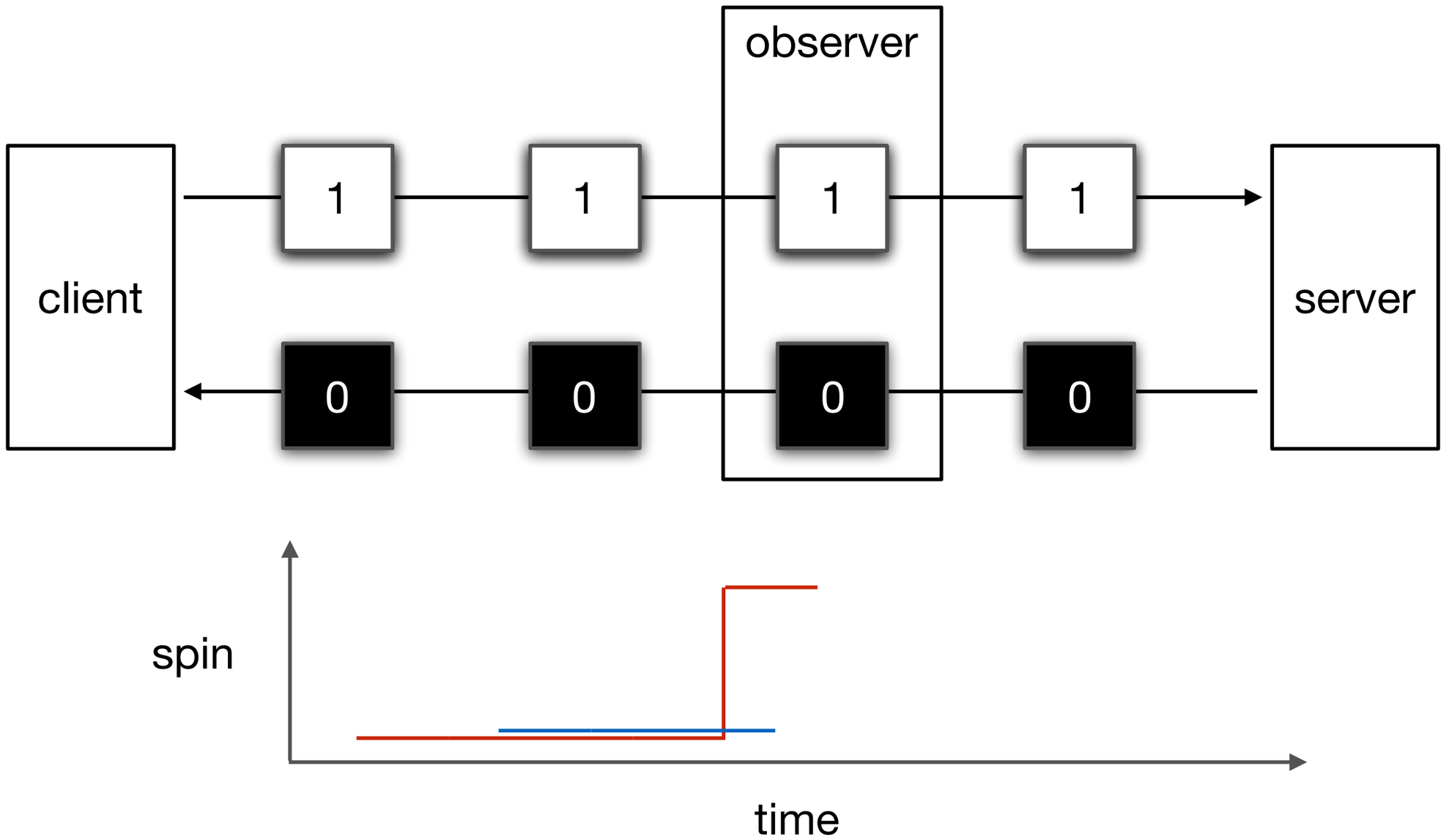
Bidirectional one-point measurement



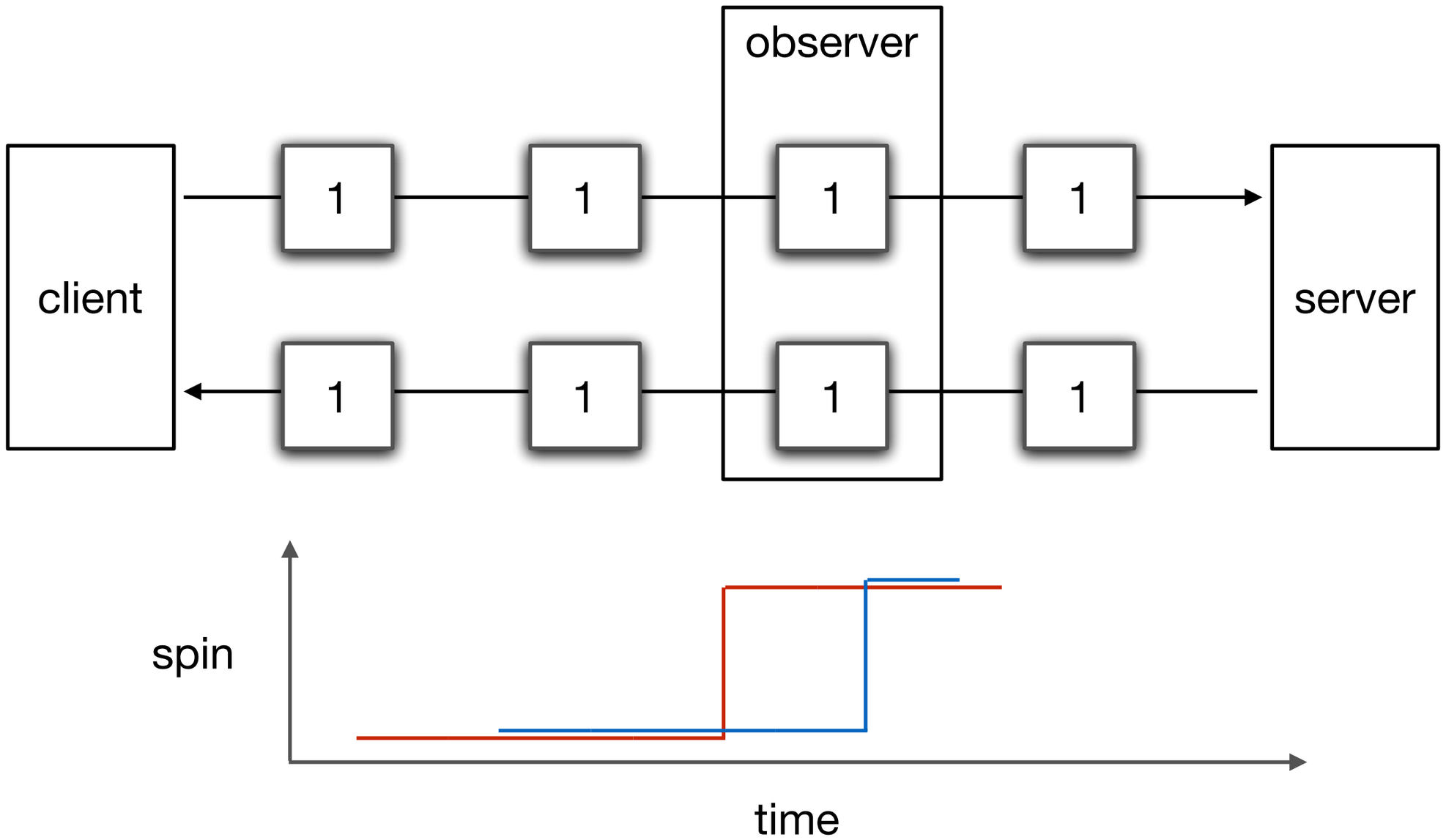
Bidirectional one-point measurement



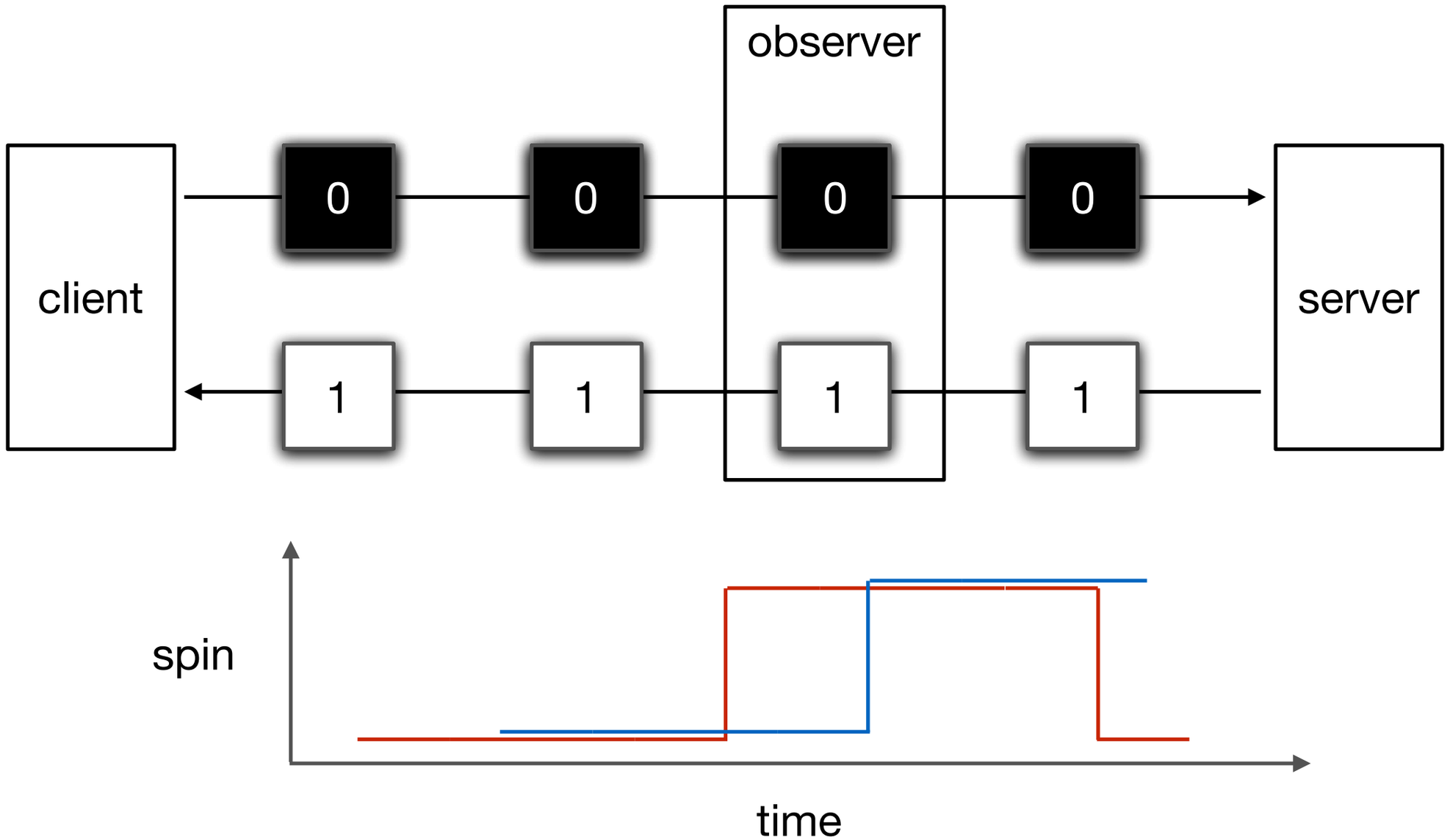
Bidirectional one-point measurement



Bidirectional one-point measurement

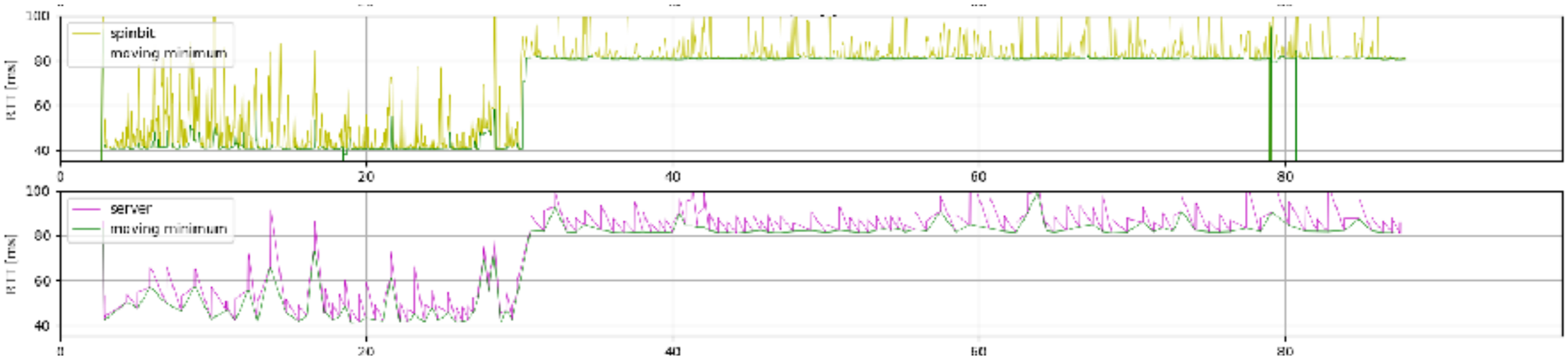


Bidirectional one-point measurement



Does it work?

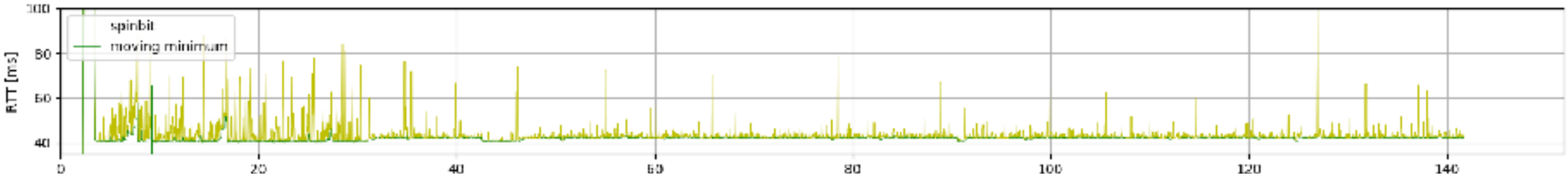
- Piet De Vaere has implemented the spin bit in minq (ekr's minimal QUIC implementation in Go)
  - Implementation effort is trivial.
- Spin signal gives high-resolution information to observers about the RTT experienced by endpoint applications.
  - Improves information available at the receiver (client) for asymmetric flows.



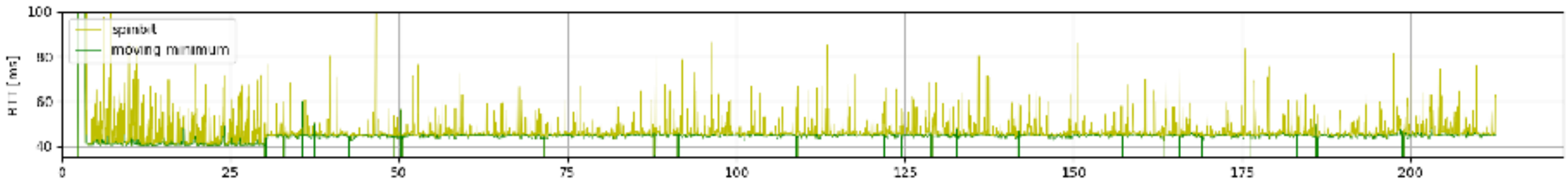
yes, it does.

Coping with Loss and Reordering

- Spin bit useful in environments in which troubleshooting signals are necessary
  - Signal survives heavy loss (~2%) with slight RTT overestimation:



  - Some loss of fidelity with heavy reordering:



- Packet numbers used to correct loss/reordering during signal generation
- Packet numbers can be used to detect loss/reordering on path *if they increment by one per packet and are in cleartext*

Fake Spin Bits?

- The spin bit can be implemented completely separate from transport mechanics; it needs only packet number information to avoid generating spurious edges.
- Why should the network trust this signal?
  - Dishonest endpoint could systematically delay or anticipate edges to generate arbitrary measured RTT values...
  - ...though this is ***trivially detectable*** by an honest endpoint.

In conclusion...

- The spin bit proposal represents a
  - minimal-overhead,
  - high-fidelity,
  - explicit signaling approach,
  - with minimal privacy impact,
- to ***replace on-path visibility*** into application-experienced RTT lost when moving from TCP (with SEQ/ACK + TSval/TSecr analysis) to QUIC.

Backup

you have questions? we have answers.

Possible enhancement: two-bit spin

- *Two-bit spin*: count 0,1,2,3,0,1,... instead of square wave
  - Server reflects, client increments by one
- Allows observers to easily cope with reordering, even with encrypted packet numbers
  - Example: reordering of the 8th and 9th packets
    - with one-bit spin: 0 0 0 0 1 1 1 0 1 0 0 0
    - with two-bit spin: 0 0 0 0 1 1 1 2 1 2 2 2
  - detected as reorder instead of as spurious edge
- Experiments show two-bit spin as good as packet numbers in rejecting reordered edges.

Possible enhancement: edge valid signal

- 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 *edge valid* bit eliminates this overestimation
  - Set when an edge is sent (more-or-less) immediately after edge value changes.
- Addition of a two-bit *edge valid* signal 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.
- Experiments show that one bit spin + two bit valid rejects invalid samples due to bursty traffic, deals with reordering as well as two-bit spin, and adds tolerance to heavy burst losses

Possible enhancement: combined spin

- Martin Thomson pointed out that two-bit spin only needs three codepoints (1,2,3), and that 0 can be used for "invalid/no signal"
    - Delayed or detected-reordered edges can be sent as a run of 0 codepoints before resuming current spin value.
- Not as good as spin + two-bit valid, but only two bits.
    - *How* not as good is a question for ongoing experimentation.

Interaction with other short header proposals

- Packet Number Encryption (#1079)
  - Packet number no longer useful for loss/reordering detection: need additional signal (e.g. spin valid) if rejecting reordered spin edges is important.
  - Type no longer necessary to encode packet number length (type bits free)
- Asymmetric Connection ID (on list, #1151): allows each side to propose a connection ID. CID is varlen, length/presence is per-flow (C bit free)