SOCKS Protocol Version 6 (update)

draft-olteanu-intarea-socks-6-05

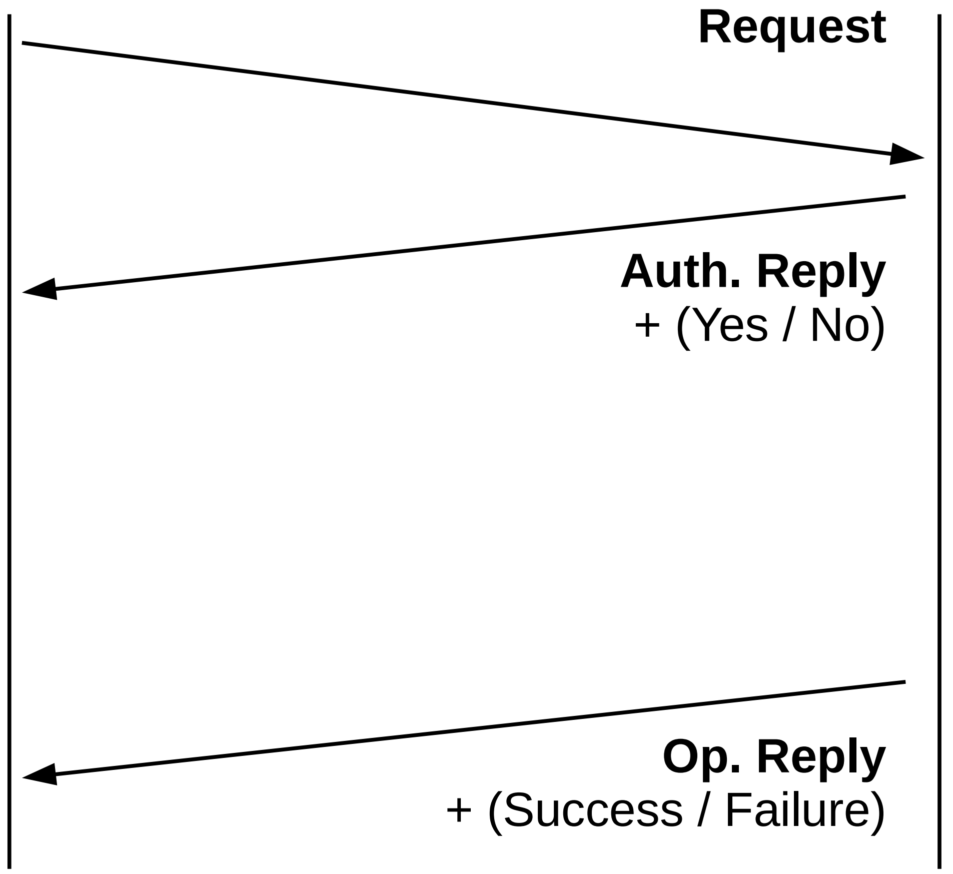
Vladimir Olteanu

New in -05

- Different handling of first bytes of application data
- Reverse TCP proxy: can now handle concurrent incoming connections to the same port
- UDP behaviour revamped

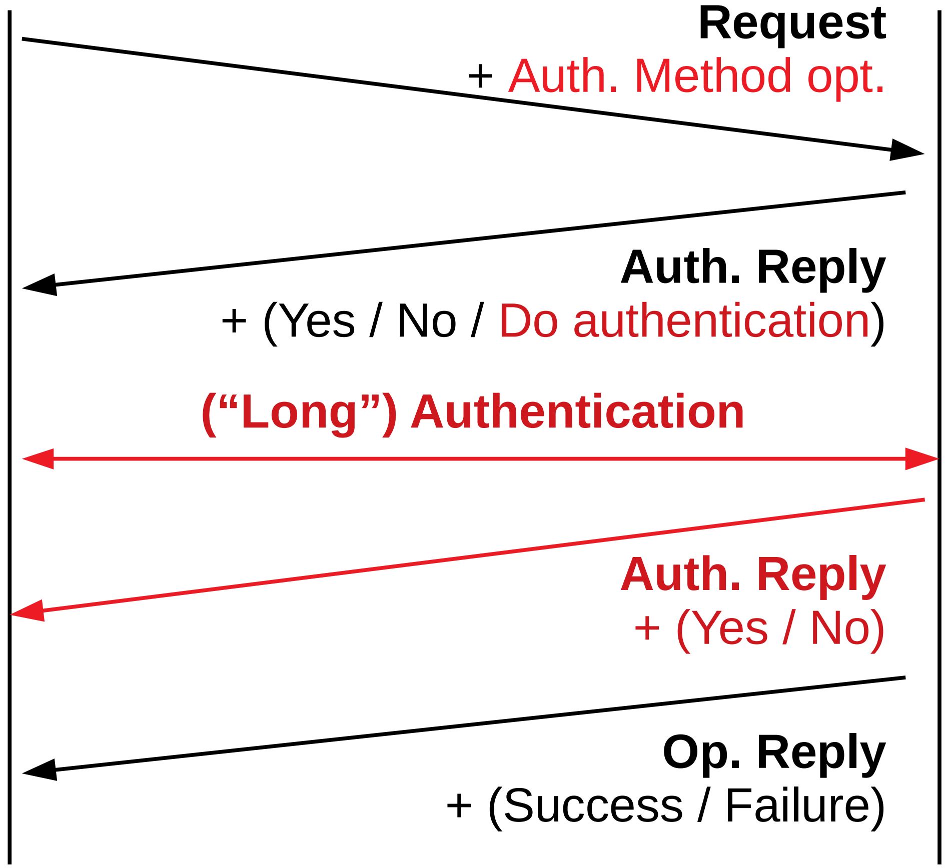
False start

- Simple core state machine



False start

- Simple core state machine
- Proxy can’t complicate it unless client asks for it



False start

- Send application data ASAP
  - Just make sure not to break the state machine

- Right after Request, if unwilling to do “long” authentication
- Right after Authentication Reply, if 0-RTT authentication succeeds
- Right after last message in authentication sequence, otherwise

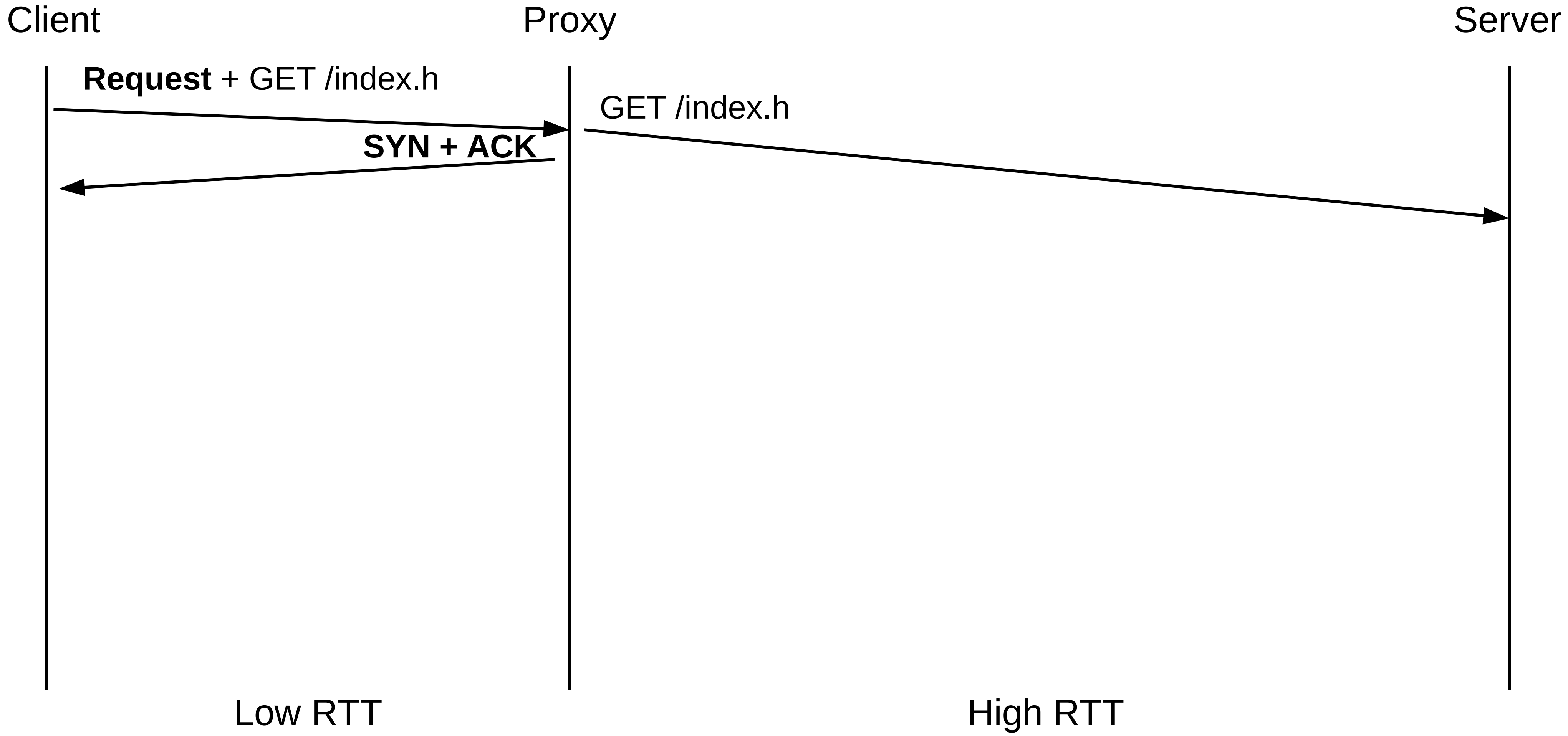
Initial data

- Serves no purpose unless “long” authentication is performed
- “Initial Data Length” field moved
  - Request → Authentication Method option
- Capped at 16K
- Can no longer be dropped by proxy
  - Removed “Initial Data Offset” field from Operation Reply

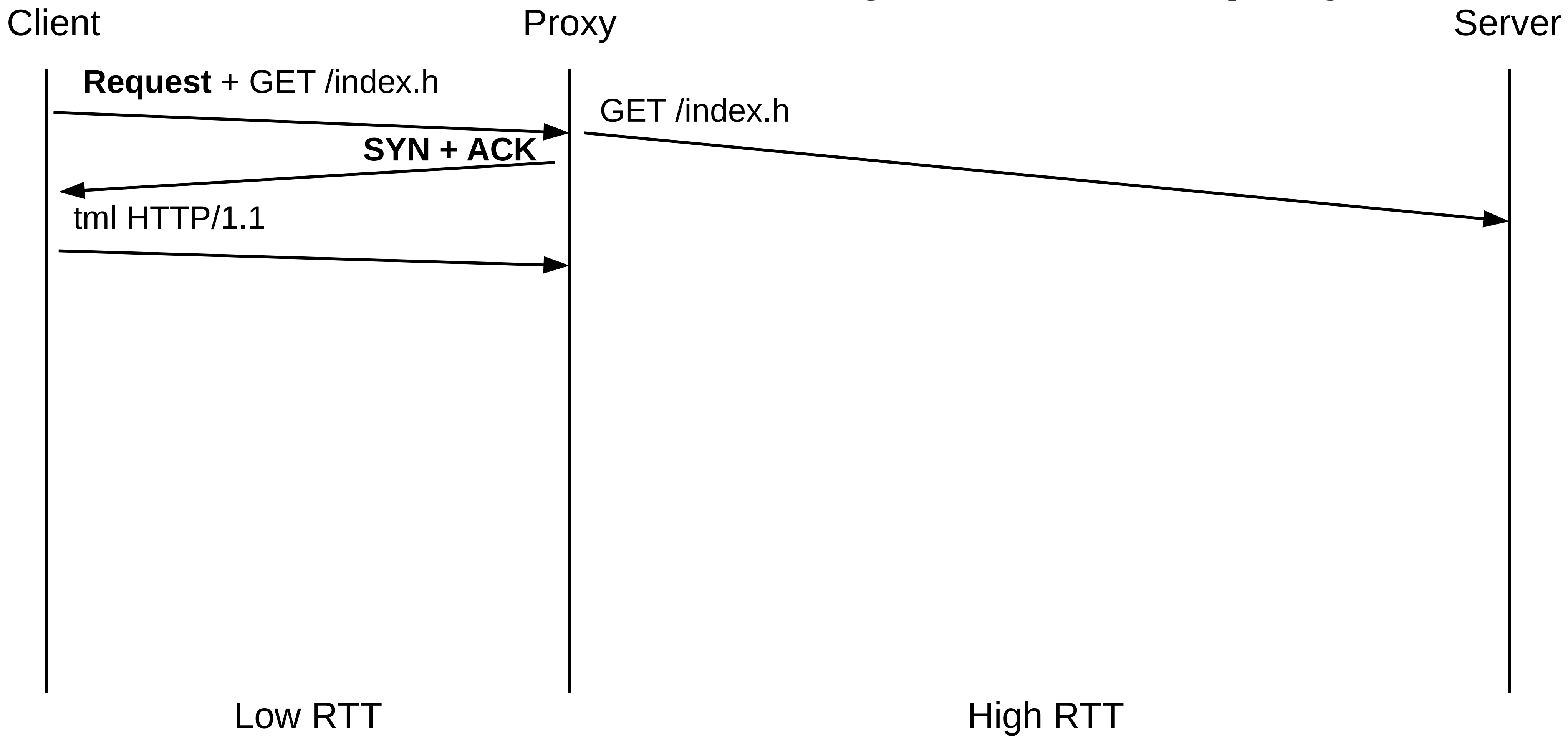
Handling TFO

- Added “Payload Length Field” to TFO Option
- Preserve TFO semantics
  - Data in TFO payload has weaker guarantees
- Ensure good timing in certain corner cases
  - Payload should be big enough to elicit a data response

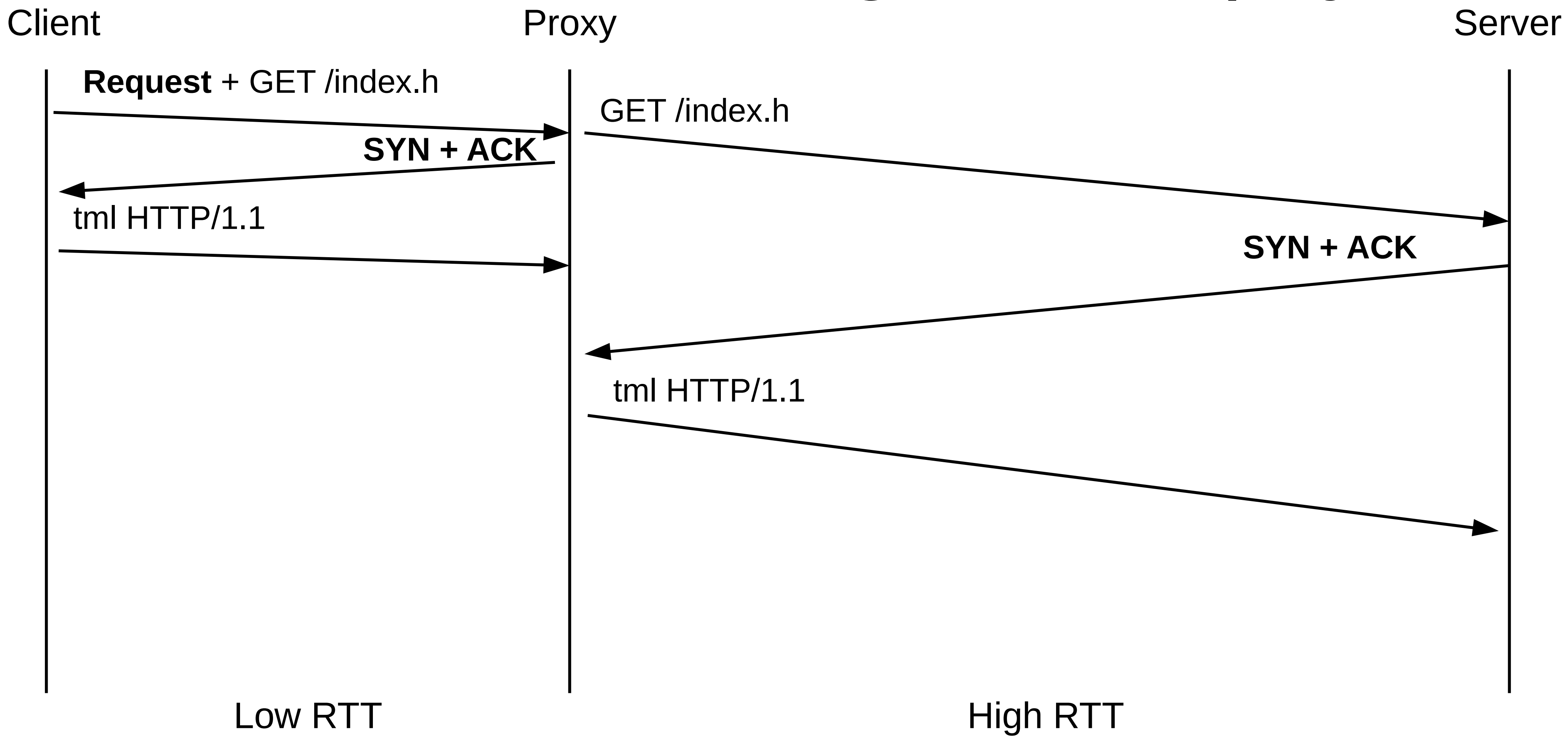
TFO corner case: fragmented payload



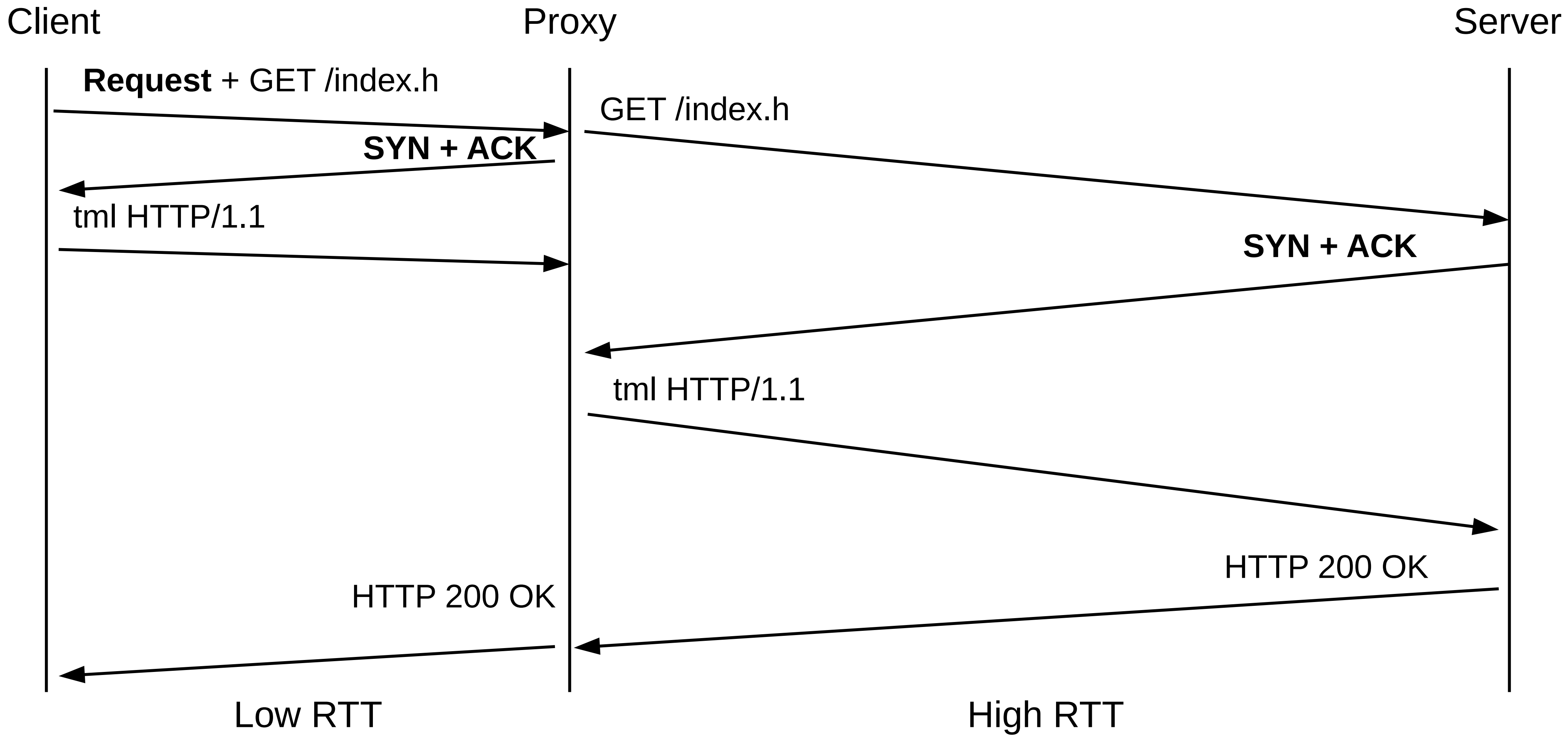
TFO corner case: fragmented payload



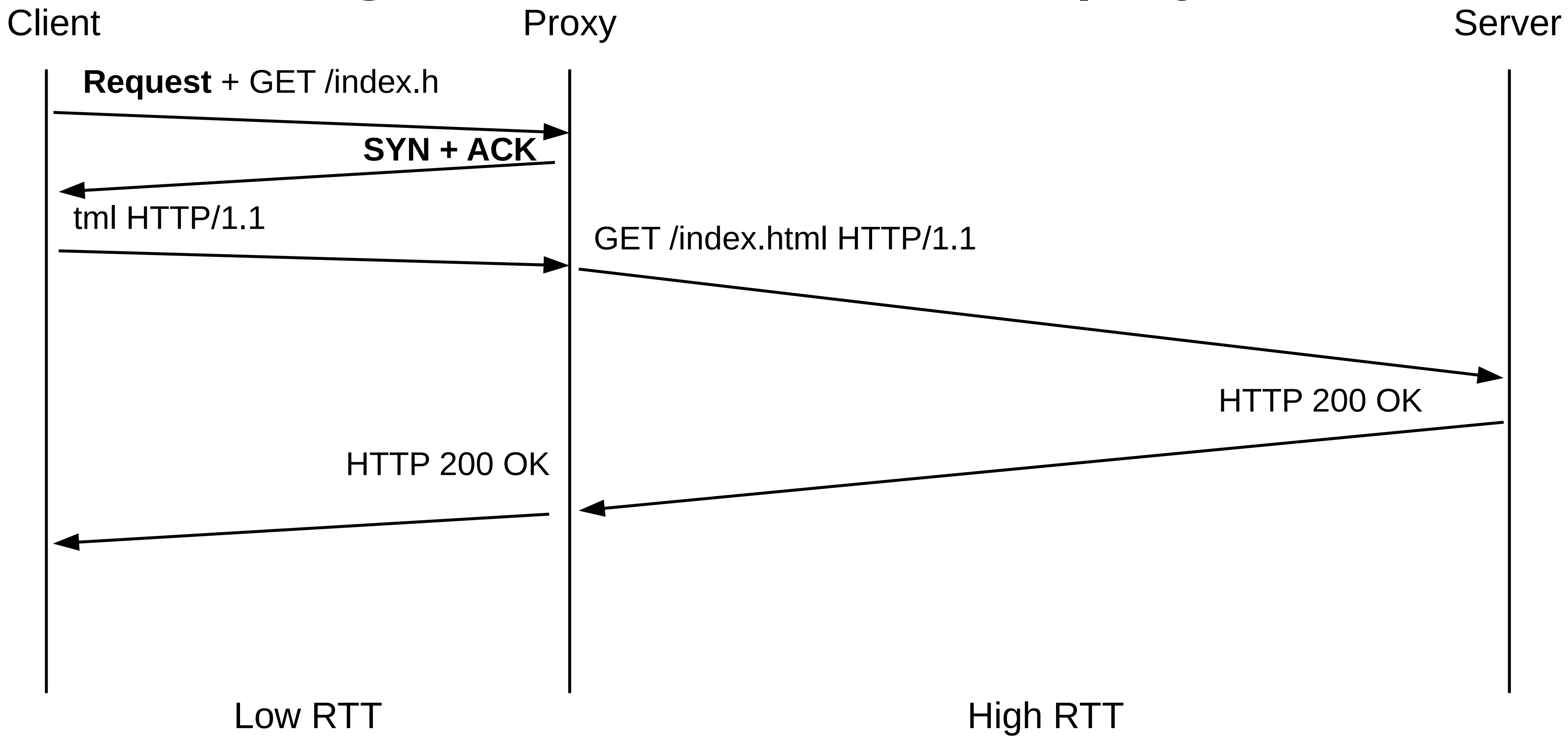
TFO corner case: fragmented payload



TFO corner case: fragmented payload



Using the correct TFO payload



TCP Reverse Proxy

- The BIND command handles one incoming connection
  - listen(), accept() once and close() listening socket
- Want to emulate typical server behavior
  - listen(), accept(), accept(), accept()…

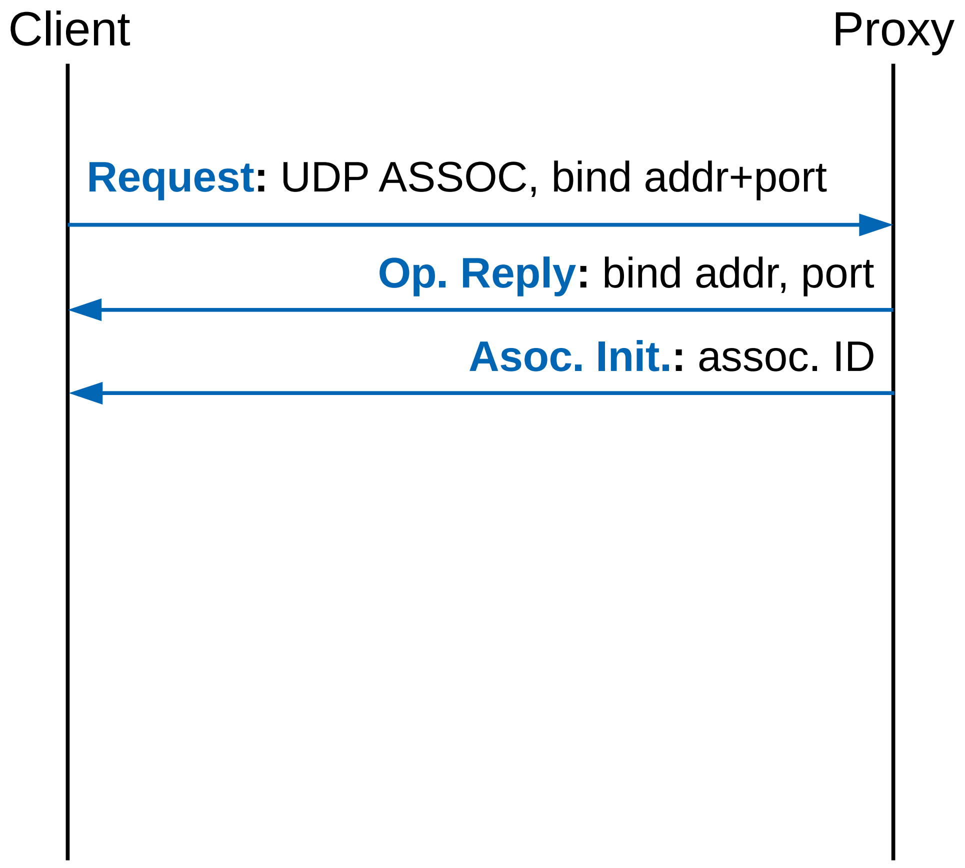
Listen Backlog Option

- First BIND: include a Listen Backlog Option
  - Prompts proxy to listen() for as long as connection is open
- Each further BIND to same address+port
  - Has the proxy accept() an incoming connection from the same listen()ing socket
- Authenticated clients only

UDP Relay

- Revamped from v5
- DTLS support
- Firewall-friendly: same relay port for all clients (1080 by default; DTLS port TBD)

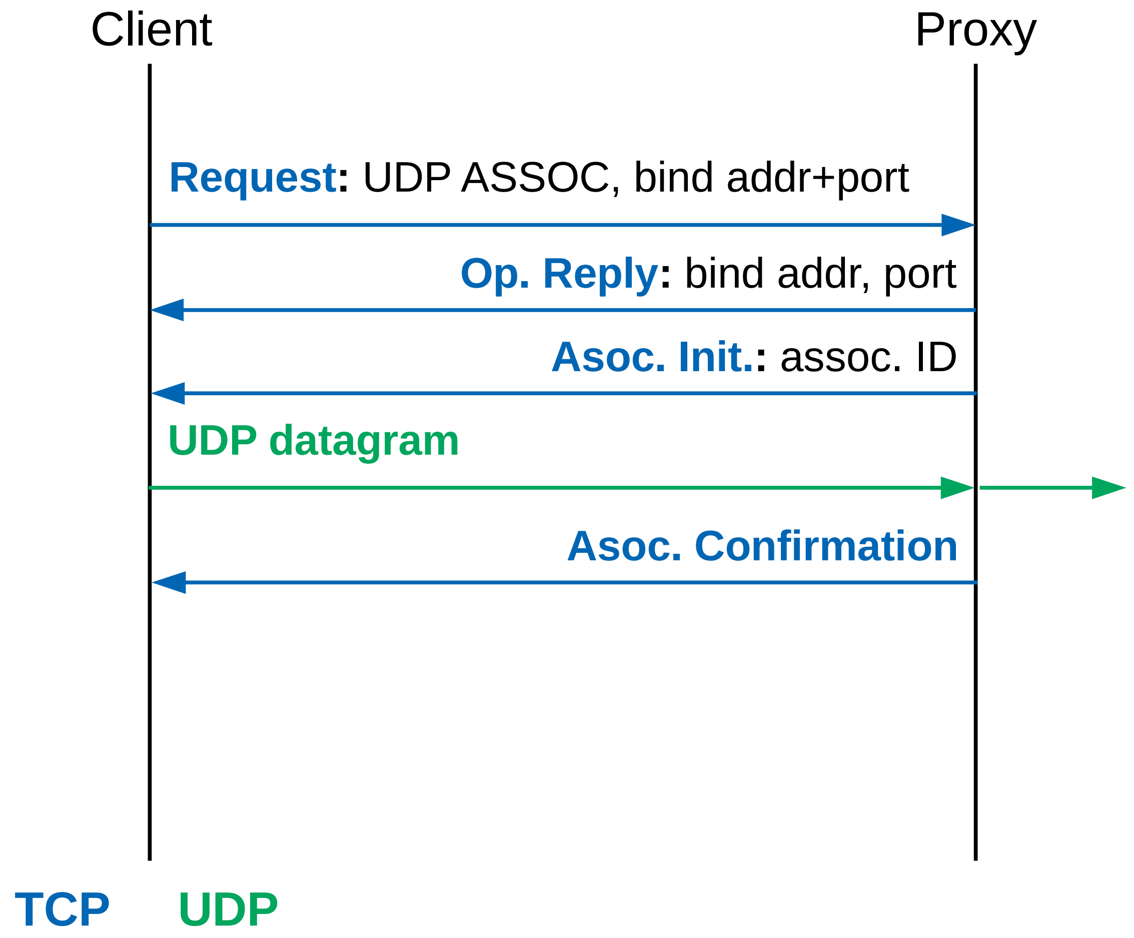
UDP Relay


Client
Proxy
Request: UDP ASSOC, bind addr+port
Op. Reply: bind addr, port
Asoc. Init.: assoc. ID

- A UDP port is bound
- An Association ID is generated for the binding

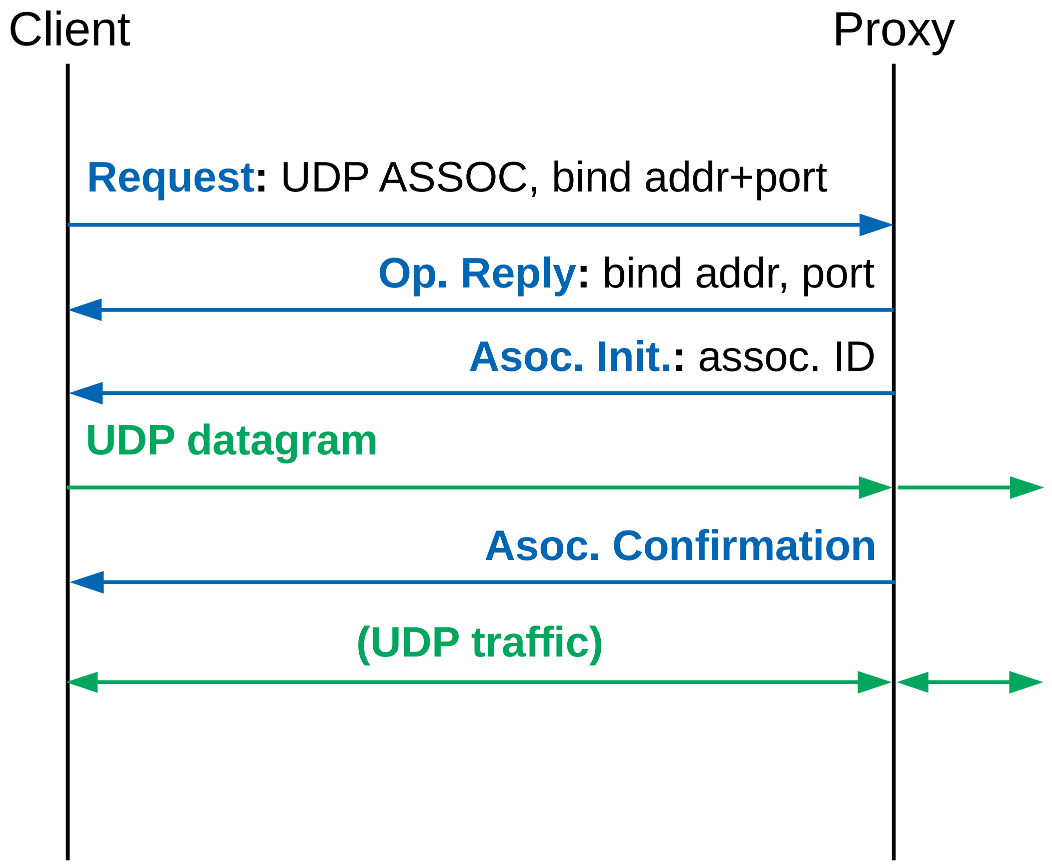
TCP UDP

UDP Relay



- The first datagram triggers an Association Confirmation
- The assoc. ID is mapped to the UDP/DTLS conversation

UDP Relay



- UDP traffic can pass in both directions now

TCP UDP

SOCKS Datagram Header

```
+---------------+-------------+------+---------+----------+
|    Version    | Association | Port | Address | Address  |
| Major | Minor |     ID      |      |  Type   |          |
+-------+-------+-------------+------+---------+----------+
|   1   |   1   |     4       |  2   |    1    | Variable |
+-------+-------+-------------+------+---------+----------+
```

- Carried by all datagrams on client-proxy leg
- Contains address of remote host
- Association ID is used for multiplexing

Nits

- TOS Stack option (useful for UDP)
- All Idempotence options now either in Requests or Authentication Replies
- Limited authentication phases to 1 (oversight)
- Removed TFO options from Operation Replies (no use case)

Implementation

- Complies with -04

- Message library: https://github.com/45G/libsocks6msg
- Utility library: https://github.com/45G/libsocks6util
- Proxyfier + proxy: https://github.com/45G/sixtysocks

What’s next?

- SOCKS Sessions
  - Killer use case: ToR (different session = different circuit)
  - Better granularity for idempotence and “multi”-bind
    - Proxy holds state per session, rather than per user