How I wrote a Python client for HTTP/3 proxies

Miloslav Pojman
@MiloslavPojman

EuroPython
Dublin, 14th July, 2022

https://pojman.cz/2022/masque/

How I wrote a Python client for HTTP/3 proxies

Miloslav Pojman
@MiloslavPojman

Miloslav Pojman

Akamai

Protocol Optimization
Prague

Akamai

360,000 servers
135 countries
250 Tbps peak traffic

Akamai

Powering and Protecting Online Privacy: iCloud Private Relay and Information for Akamai Customers

https://www.akamai.com/blog/cloud/powering-and-protecting-online-privacy-icloud-private-relay

iCloud Private Relay Overview

Learn how Private Relay protects users' privacy on the internet.

December 2021

https://www.apple.com/privacy/docs/iCloud_Private_Relay_Overview_Dec2021.PDF

MASQUE

Multiplexed Application Substrate over QUIC Encryption

https://tools.ietf.org/id/draft-schinazi-masque-01.html

Photo by Jean-Philippe Delberghe on Unsplash

HTTP/3 Proxy

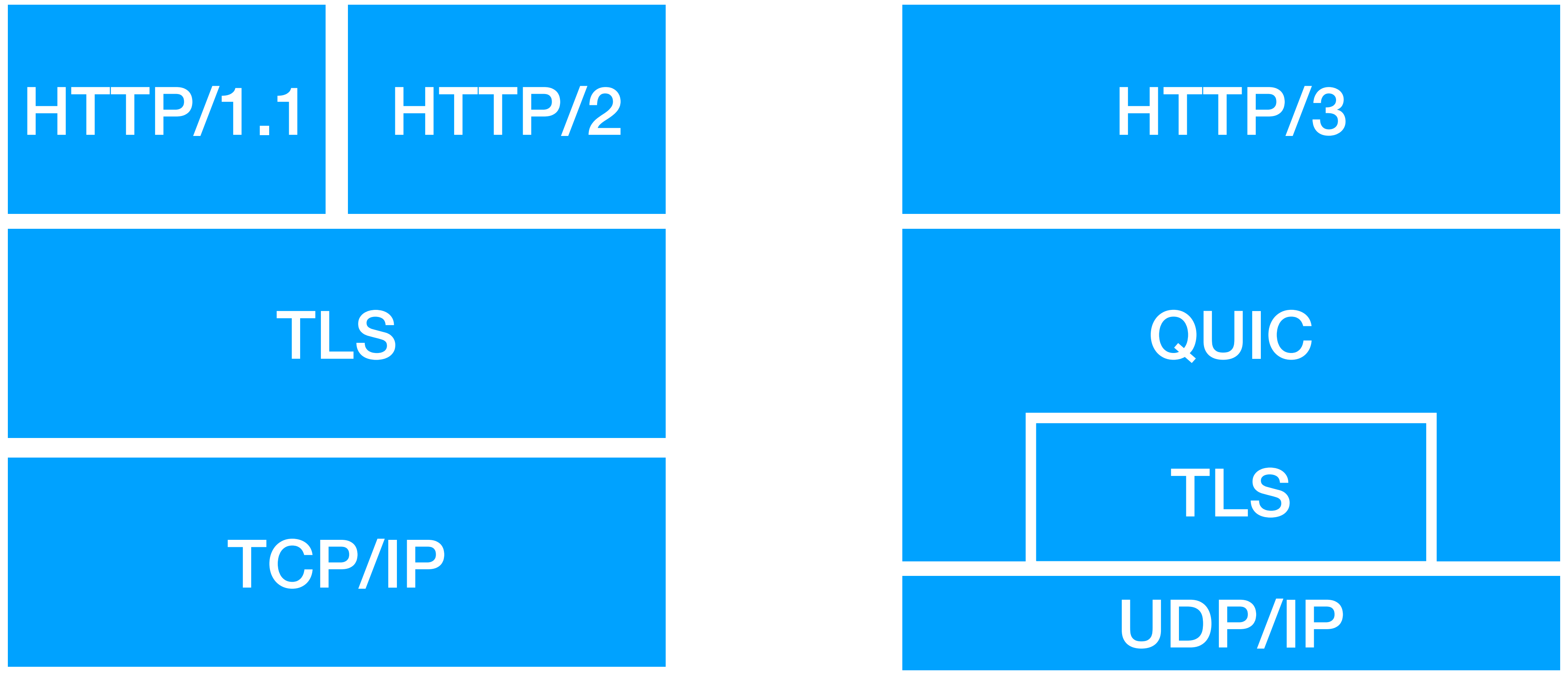
Tunnels

Photo by Andre Benz on Unsplash

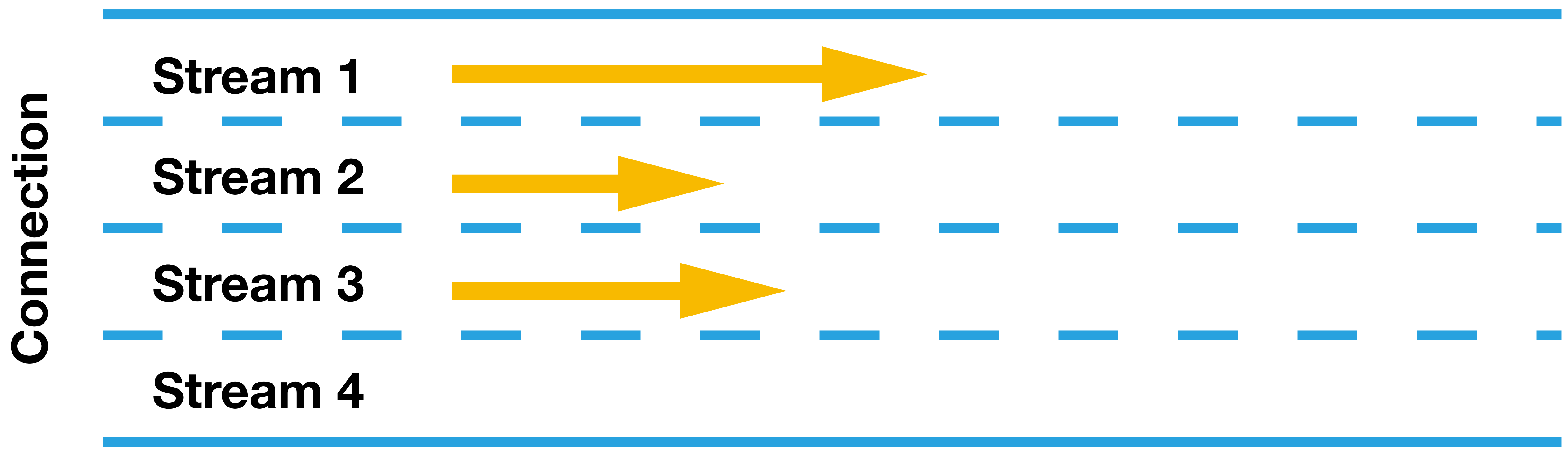
HTTP/3 = HTTP over QUIC

- UDP
- Multiplexed
- Fully encrypted

QUIC = an alternative to TCP



Multiplexing



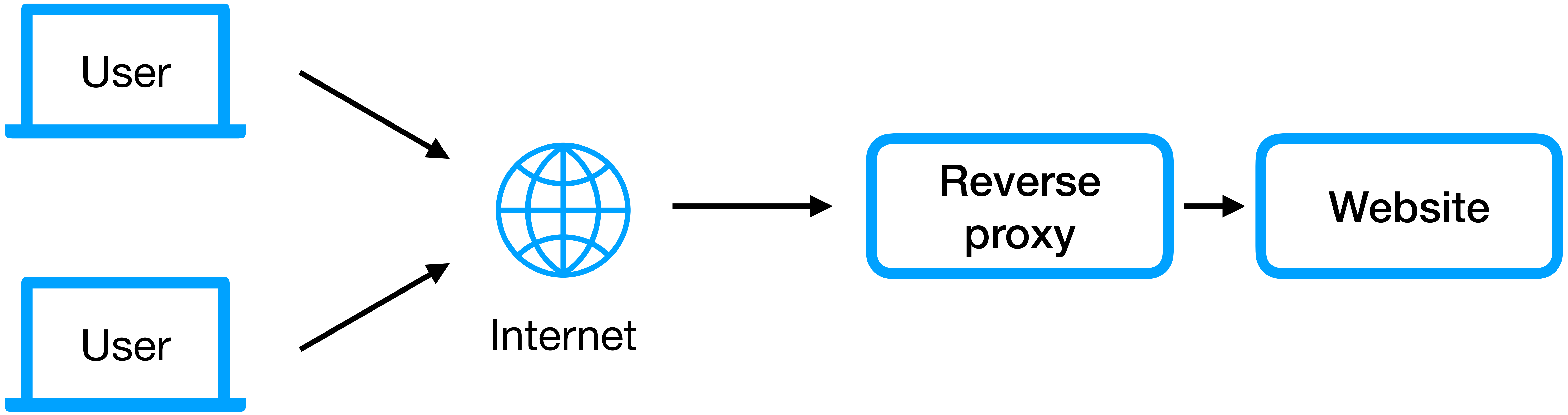
HTTP/2 - at the HTTP layer
HTTP/3 - at the QUIC layer

Proxies

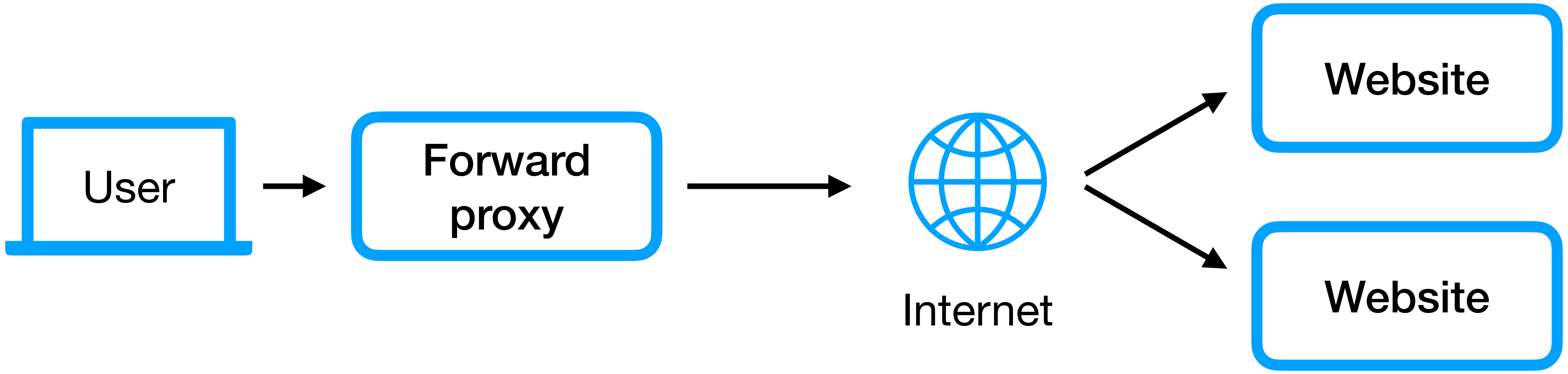
Forward vs reverse

Photo by Alok Sharma on Unsplash

Reverse proxies



Forward proxies

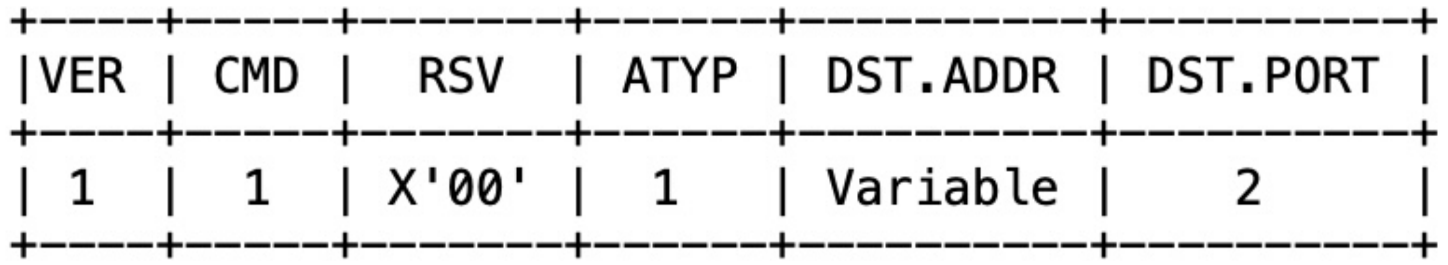


HTTP vs. SOCKS

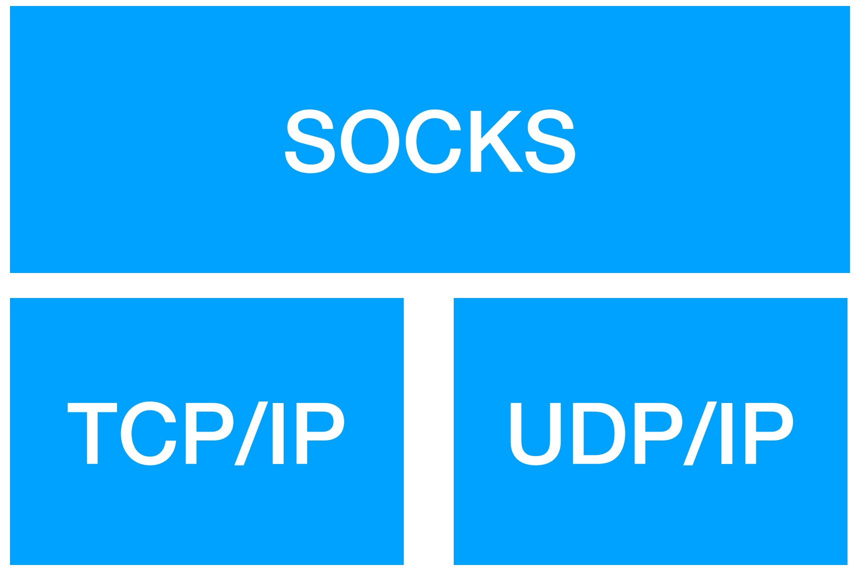
Photo by Christian Fickinger on Unsplash

SOCKS proxy

```
ssh -N -D 1080 $PROXY
```



VER	CMD	RSV	ATYP	DST.ADDR	DST.PORT
1	1	X'00'	1	Variable	2

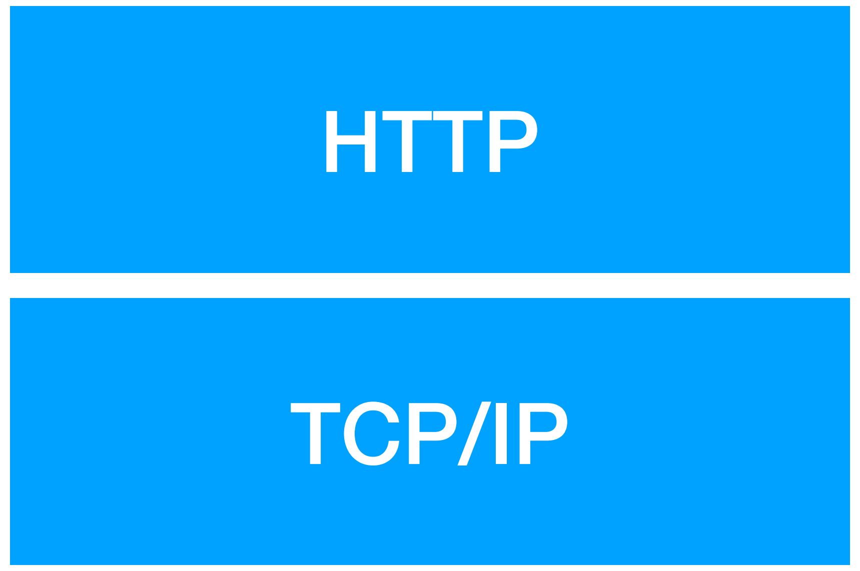


https://datatracker.ietf.org/doc/html/rfc1928

HTTP proxy

```
git clone https://github.com/urllib3/urllib3
python3 -m dummyserver.proxy
```

HTTP



HTTP

TCP/IP

https://www.rfc-editor.org/rfc/rfc9110
https://www.rfc-editor.org/rfc/rfc9111
https://www.rfc-editor.org/rfc/rfc9112

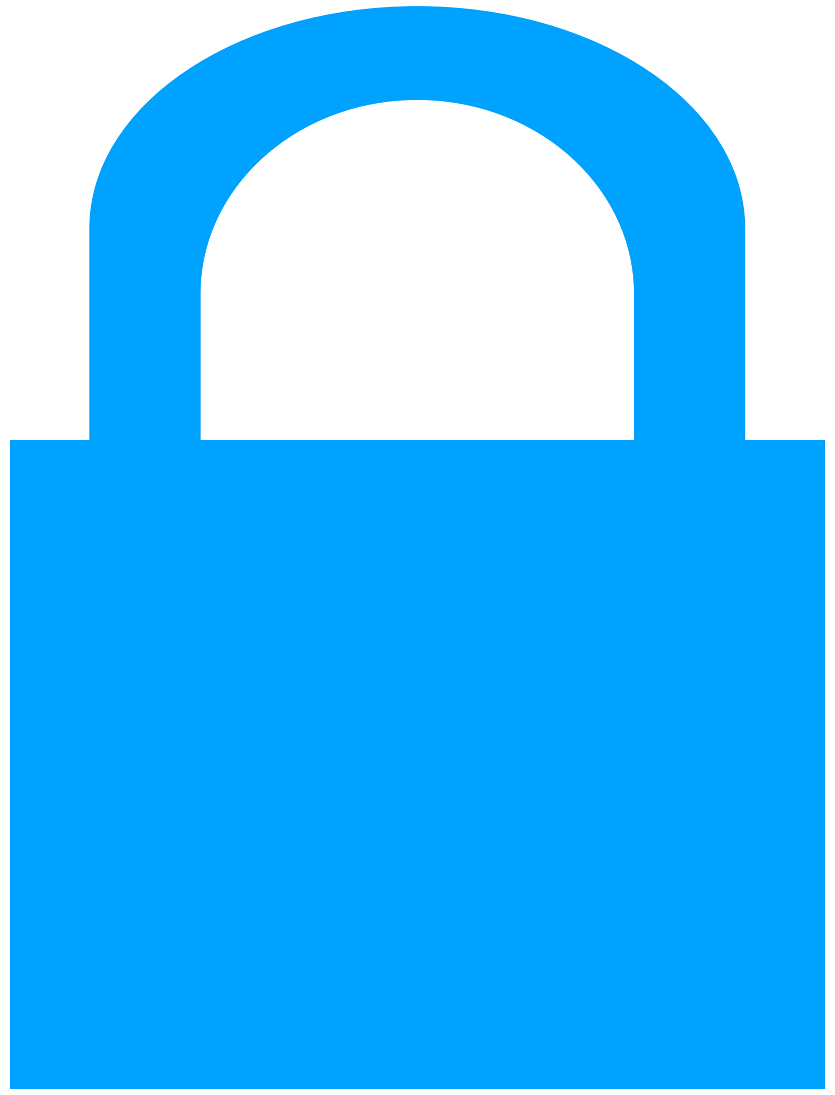
HTTP forward proxies

HTTP request

```
$ nc example.com 80
GET / HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Accept-Ranges: bytes
...
Content-Length: 1256

<!doctype html>
<html>
<head>
...
```

Proxy forwarding

```
$ nc localhost 8888
GET http://example.com/ HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
Accept-Ranges: bytes
...
Content-Length: 1256

<!doctype html>
<html>
<head>
...
```

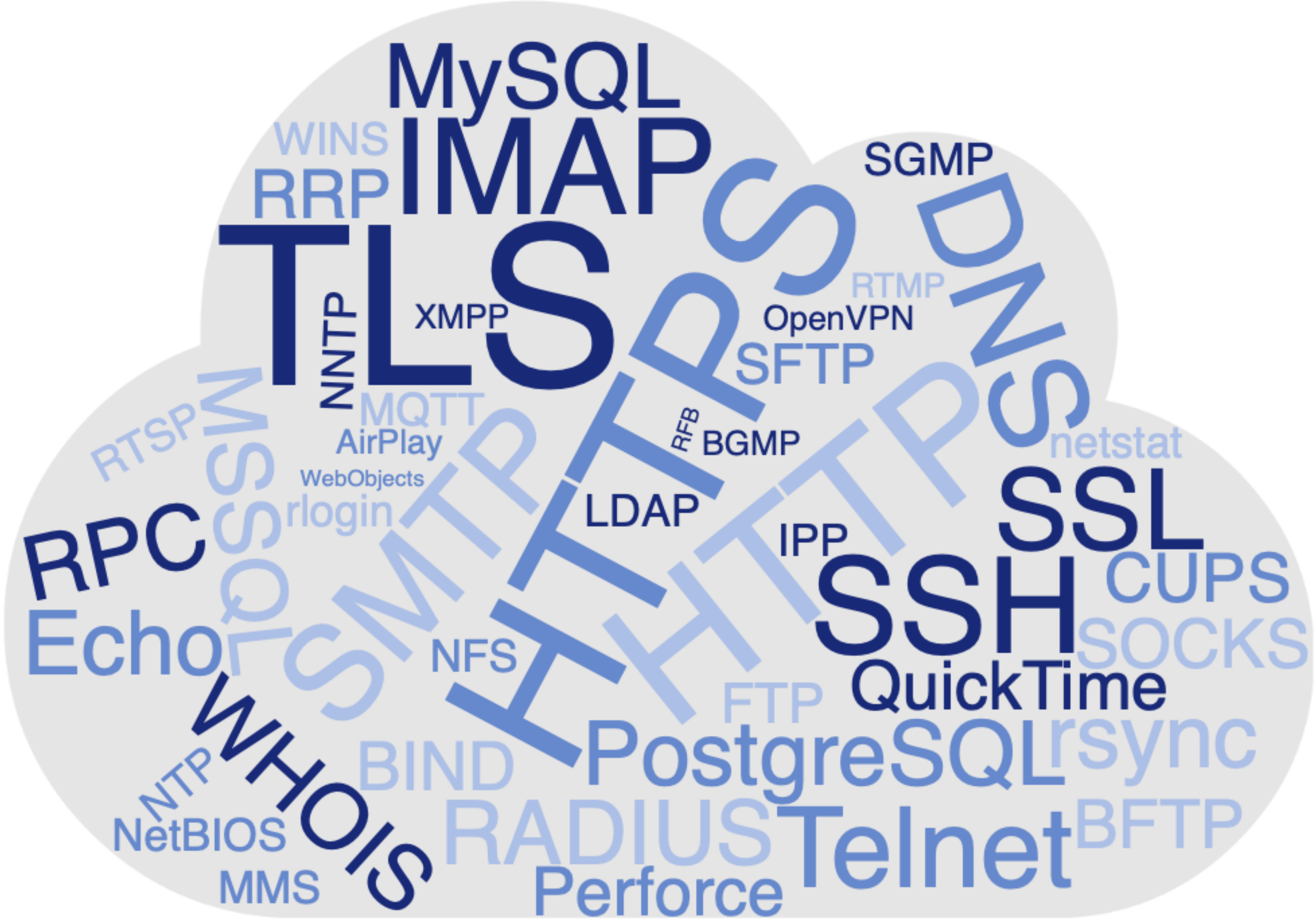


HTTPS

HTTP over TLS

Proxy tunneling

```
$ nc localhost 8888
CONNECT example.com:80 HTTP/1.1

HTTP/1.1 200 Connection established

GET http://example.com/ HTTP/1.1

HTTP/1.1 200 OK
...
```



Tunneling

Protocol agnostic

Onion routing

```
$ nc proxy1 80
CONNECT proxy2:80 HTTP/1.1

HTTP/1.1 200 Connection established

CONNECT proxy3:80 HTTP/1.1

HTTP/1.1 200 Connection established

CONNECT ...
```

Photo by Andre Benz on Unsplash

MASQUE

- HTTP/3 with HTTP/2 fallback
- Tunneling mode only
- Onion routing
- DNS over HTTPS, UDP Proxying, QUIC Proxying, …

HTTP clients

Python HTTP clients

High-level	urllib.request	requests	httpx
Low-level	http.client	urllib3	httpcore
	HTTP/1.1	**HTTP/1.1**	**HTTP/1.1** **HTTP/2**

Sans IO

h11

h2

aioquic

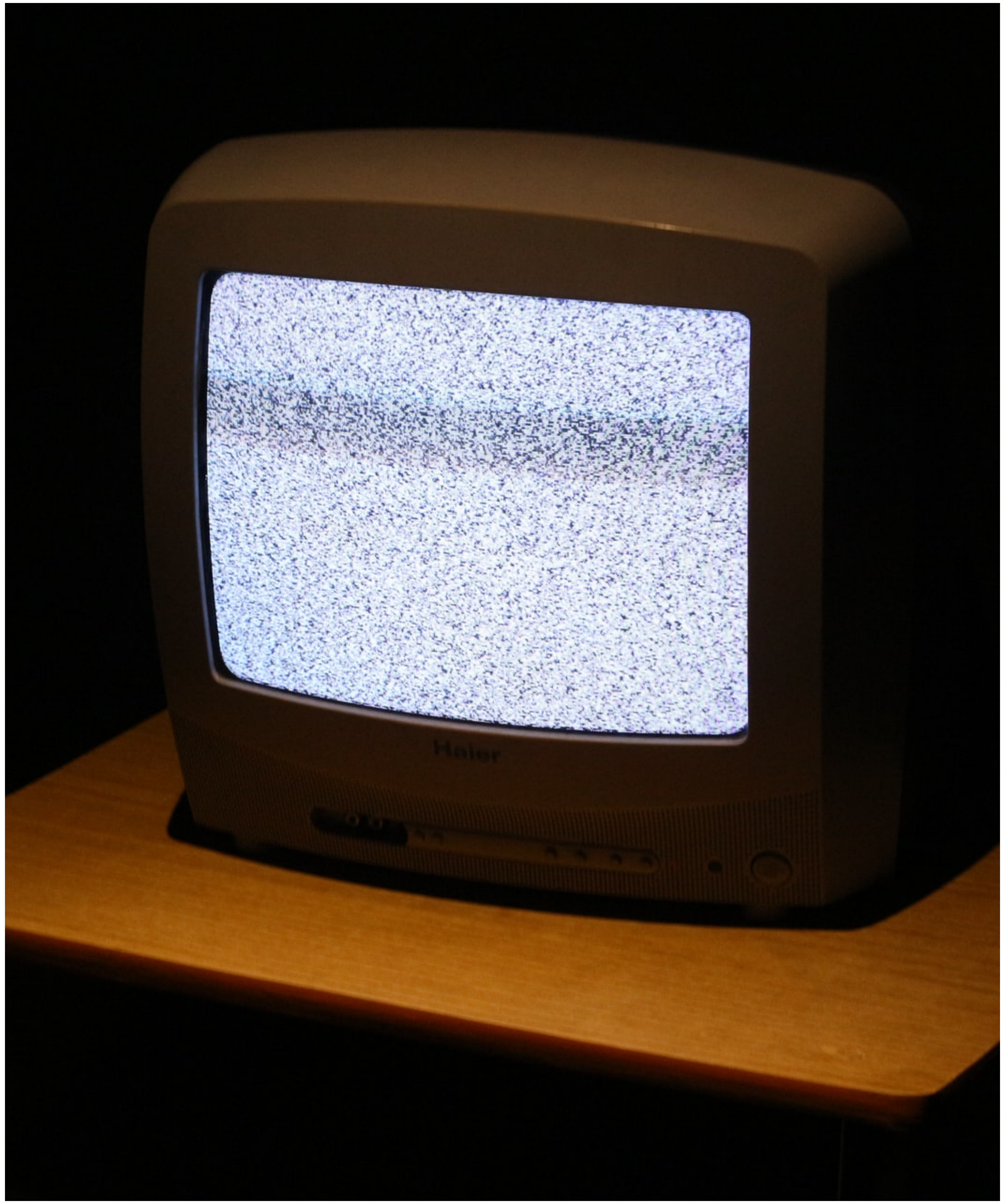


Photo by aj_aaaab on Unsplash

Compiled implementations

nghttp2

google/quiche
cloudflare/quiche
msquic

Photo by David Behar on Unsplash

Proxy

none

SOCKS

HTTP(S) forwarding

HTTP(S) tunneling

MASQUE

×

Origin

HTTP/1.1

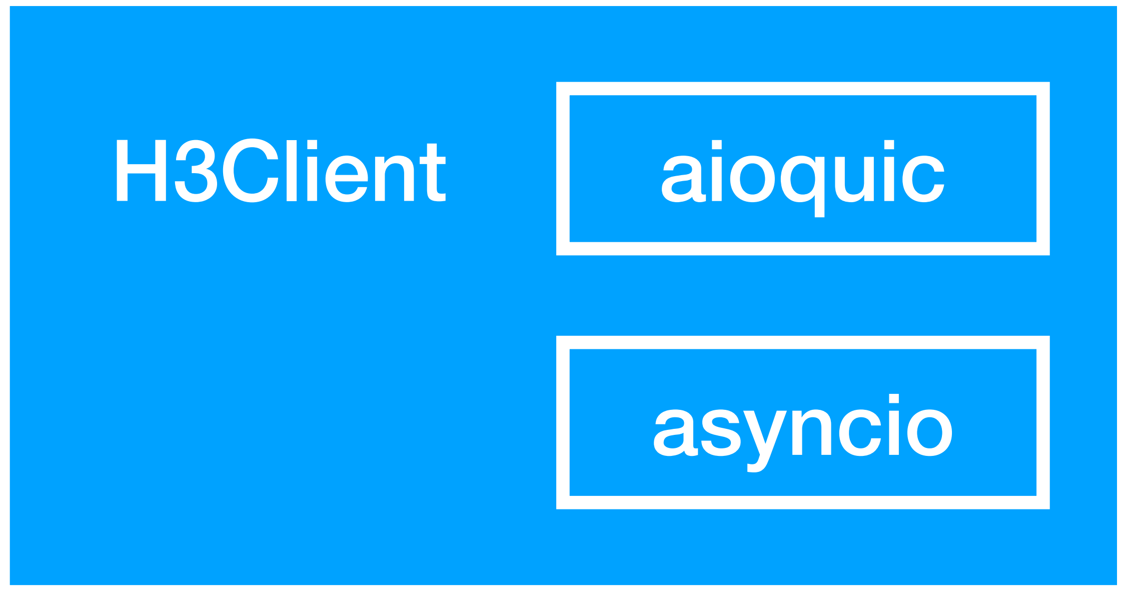
HTTP/1.1 over TLS

HTTP/2

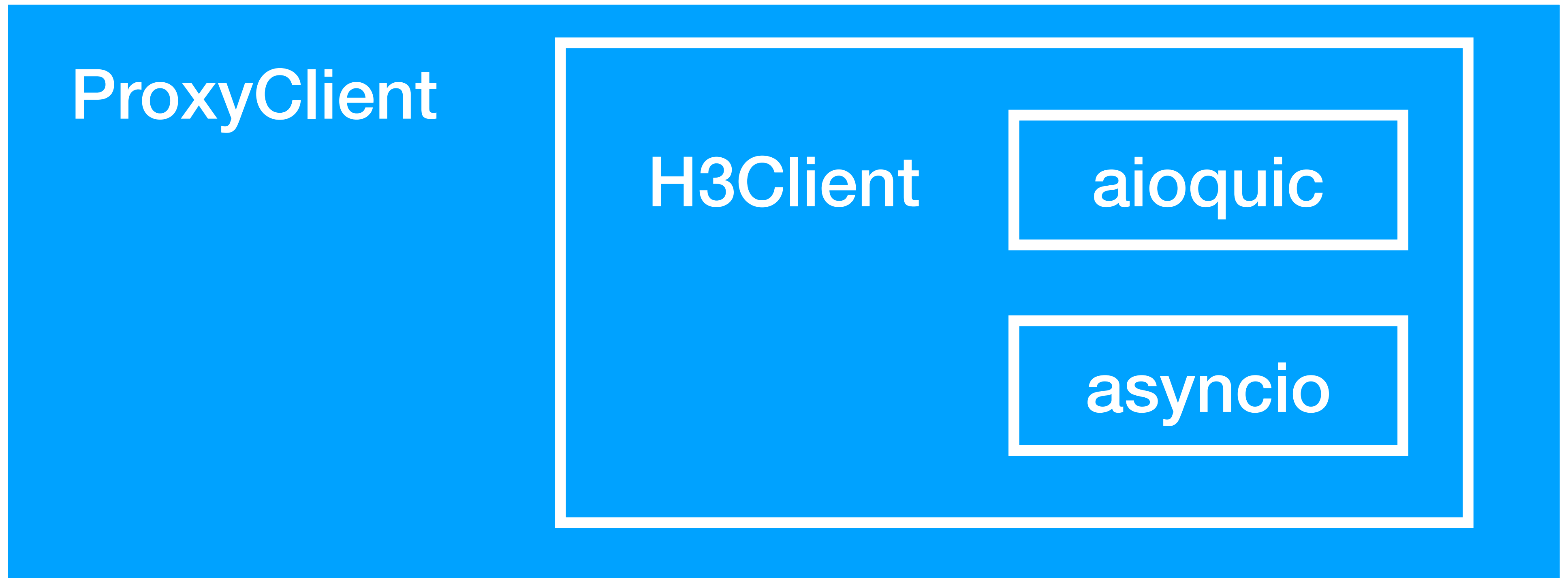
HTTP/3

My client

H3 client



Proxy client

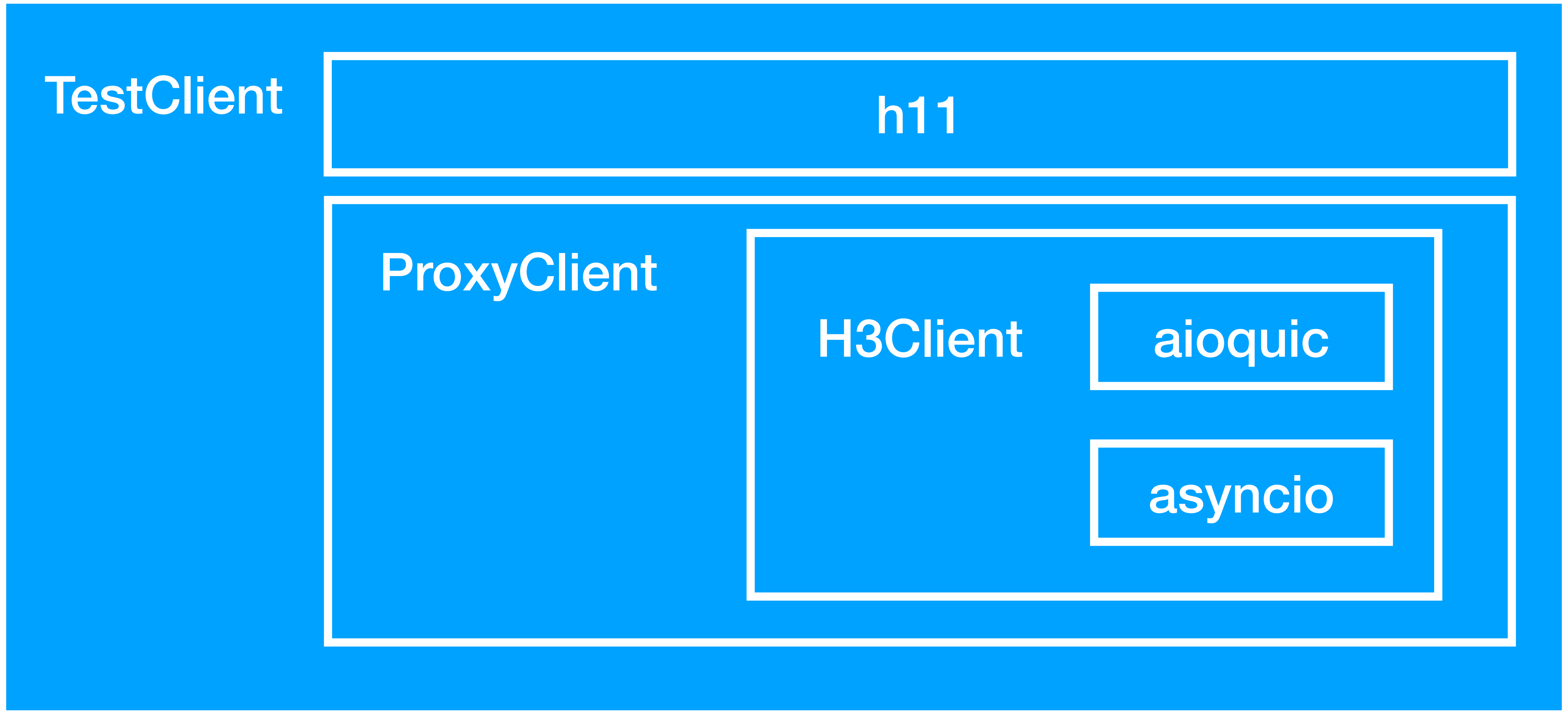

ProxyClient

H3Client

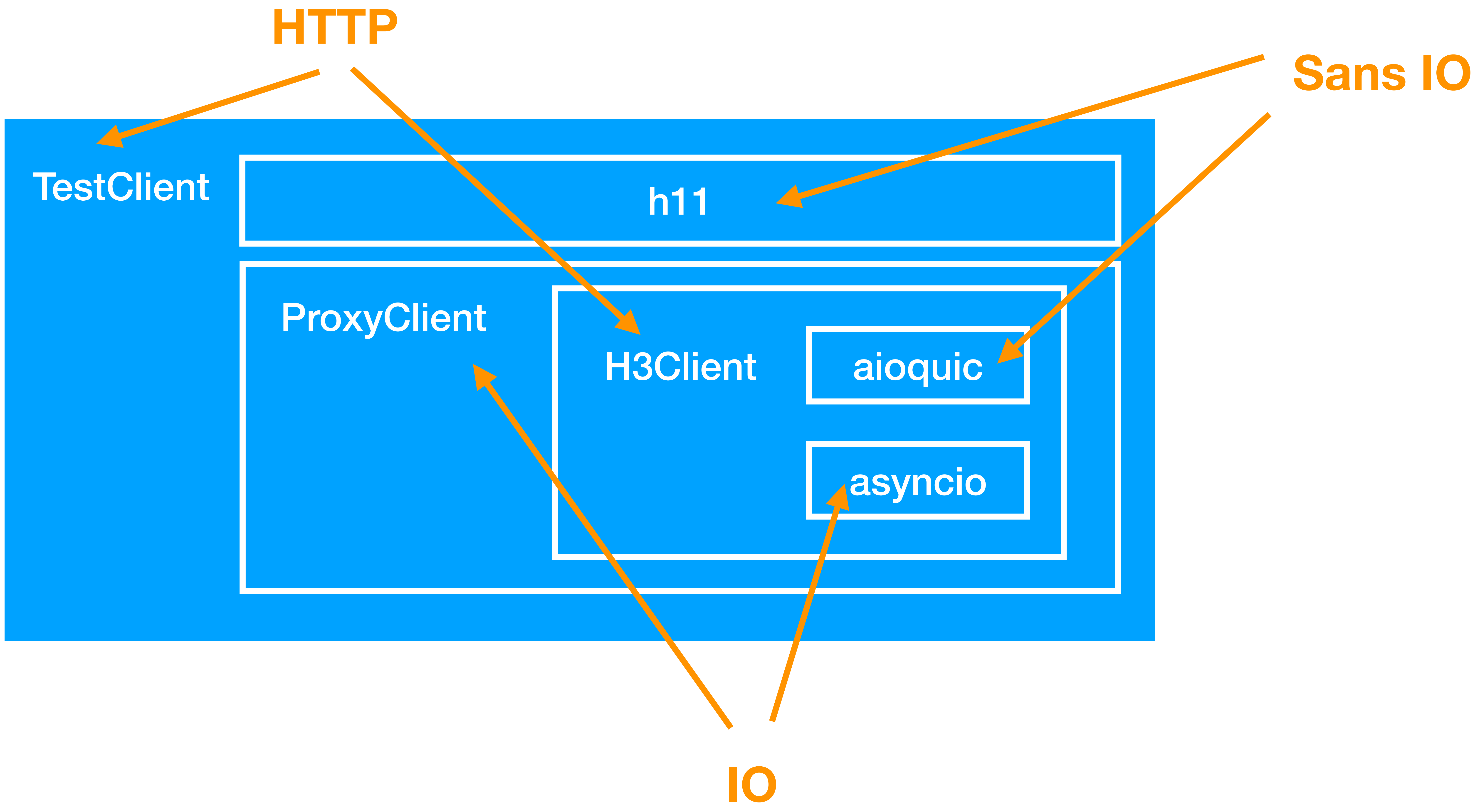
aioquic

asyncio

Test client

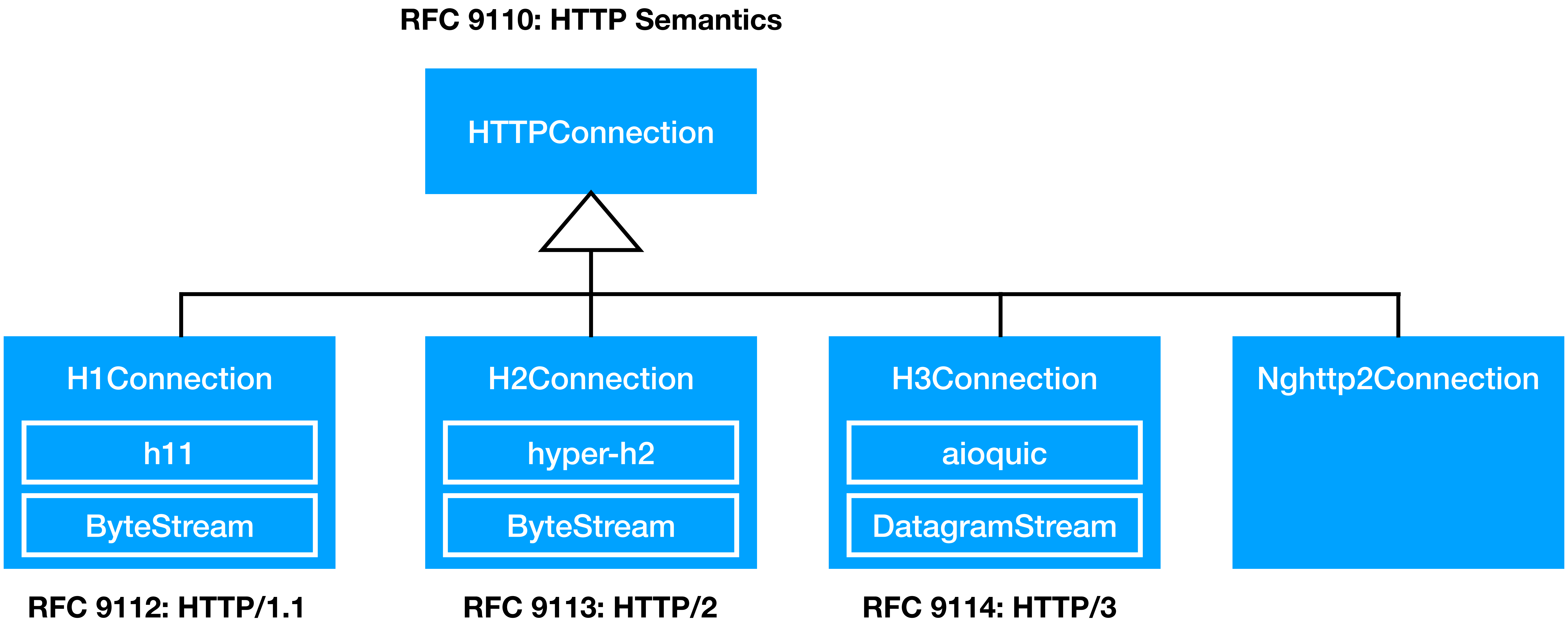


Pattern?


HTTP
Sans IO
TestClient
h11
ProxyClient
H3Client
aioquic
asyncio
IO

Vision

HTTP interface



RFC 9110: HTTP Semantics

HTTPConnection

H1Connection

h11

ByteStream

RFC 9112: HTTP/1.1

H2Connection

hyper-h2

ByteStream

RFC 9113: HTTP/2

H3Connection

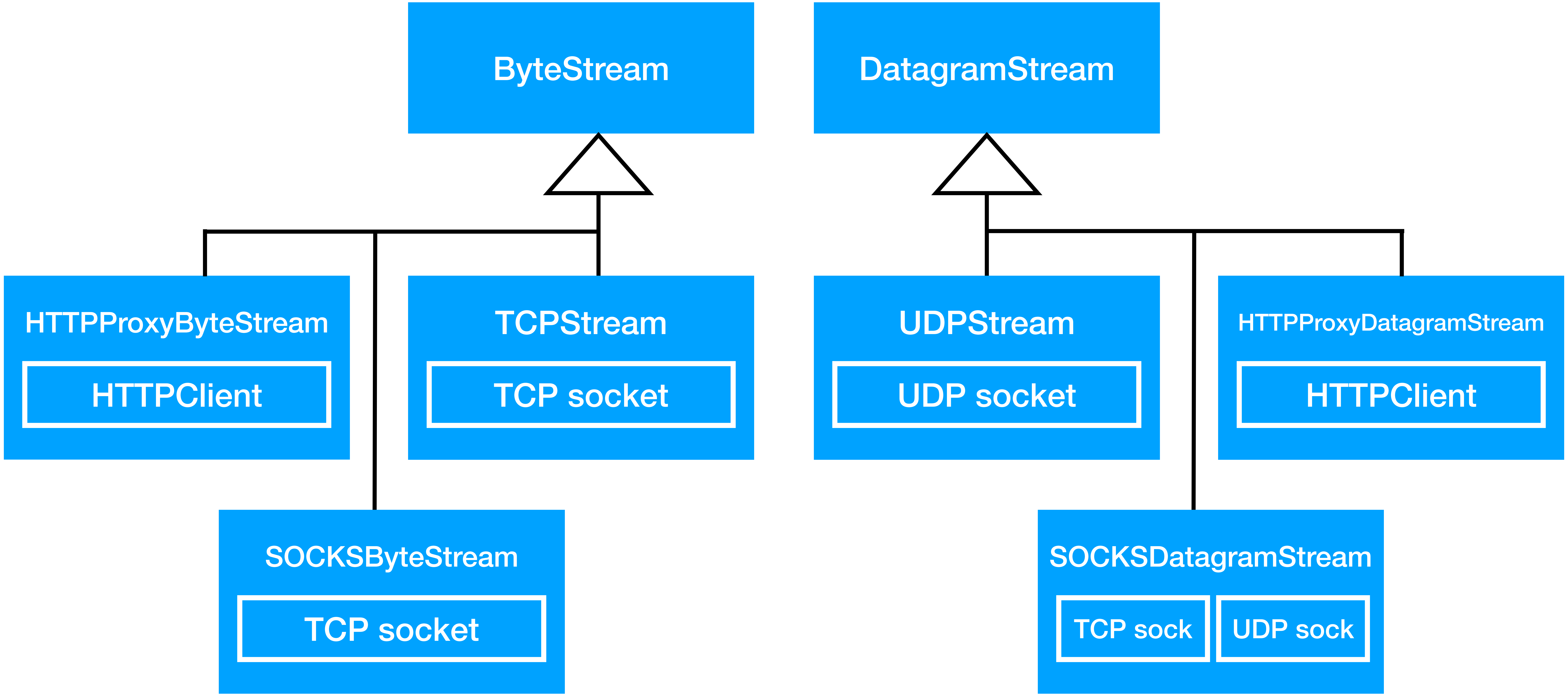
aioquic

DatagramStream

RFC 9114: HTTP/3

Nghttp2Connection

Network interfaces

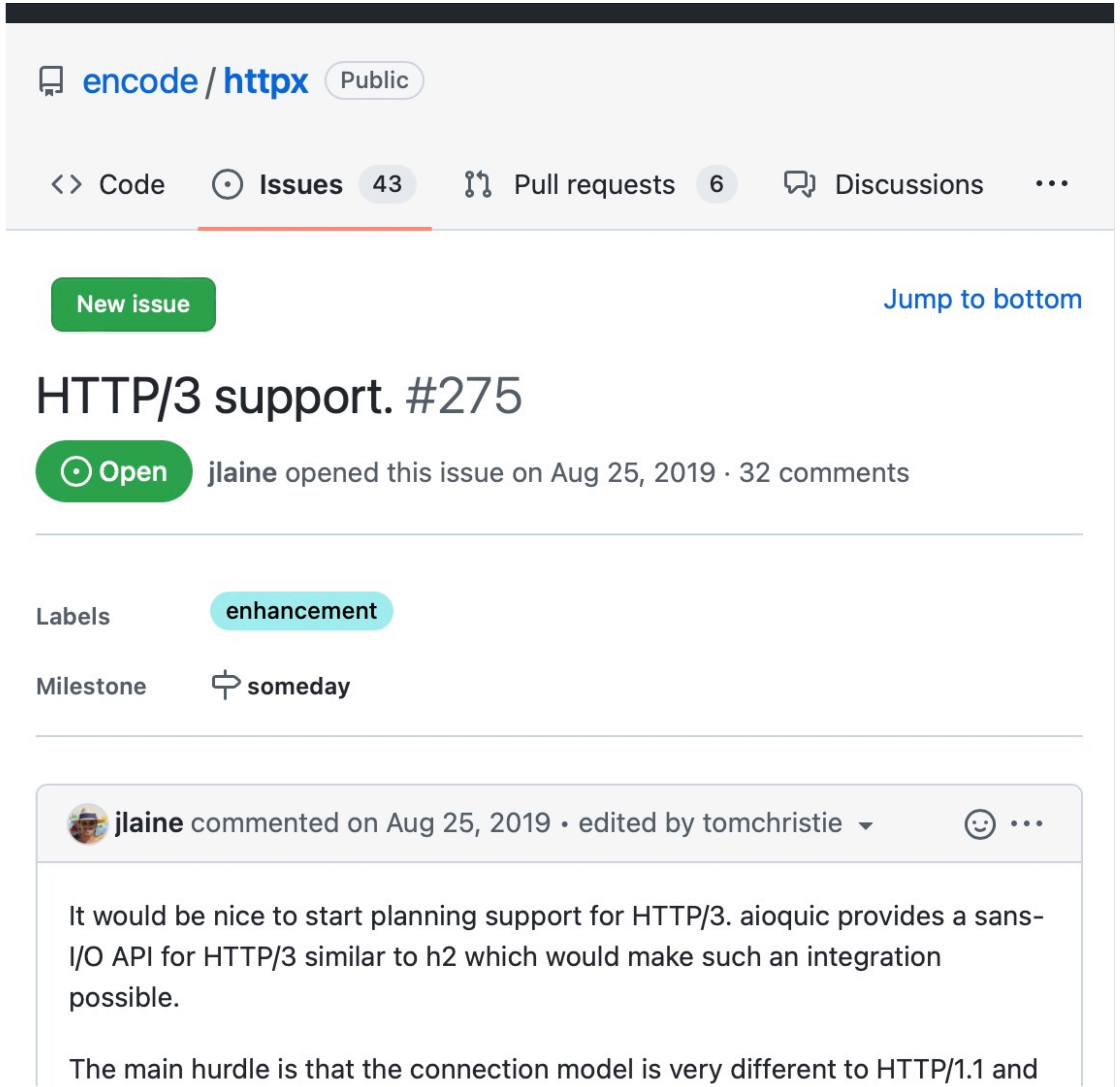


Design considerations

- Multiplexing → `async`
- QUIC → `async with QUICConnection(…)`
- asyncio event driven protocols vs. trio (anyio) streams

Contributing?

- HTTP/3 support
- Multiple protocols for proxy connections
- Changes to internals of low-level httpcore



HTTP/3 support. #275

Open jlaine opened this issue on Aug 25, 2019 · 32 comments

Labels enhancement

Milestone someday

jlaine commented on Aug 25, 2019 • edited by tomchristie

It would be nice to start planning support for HTTP/3. aioquic provides a sans-I/O API for HTTP/3 similar to h2 which would make such an integration possible.

The main hurdle is that the connection model is very different to HTTP/1.1 and

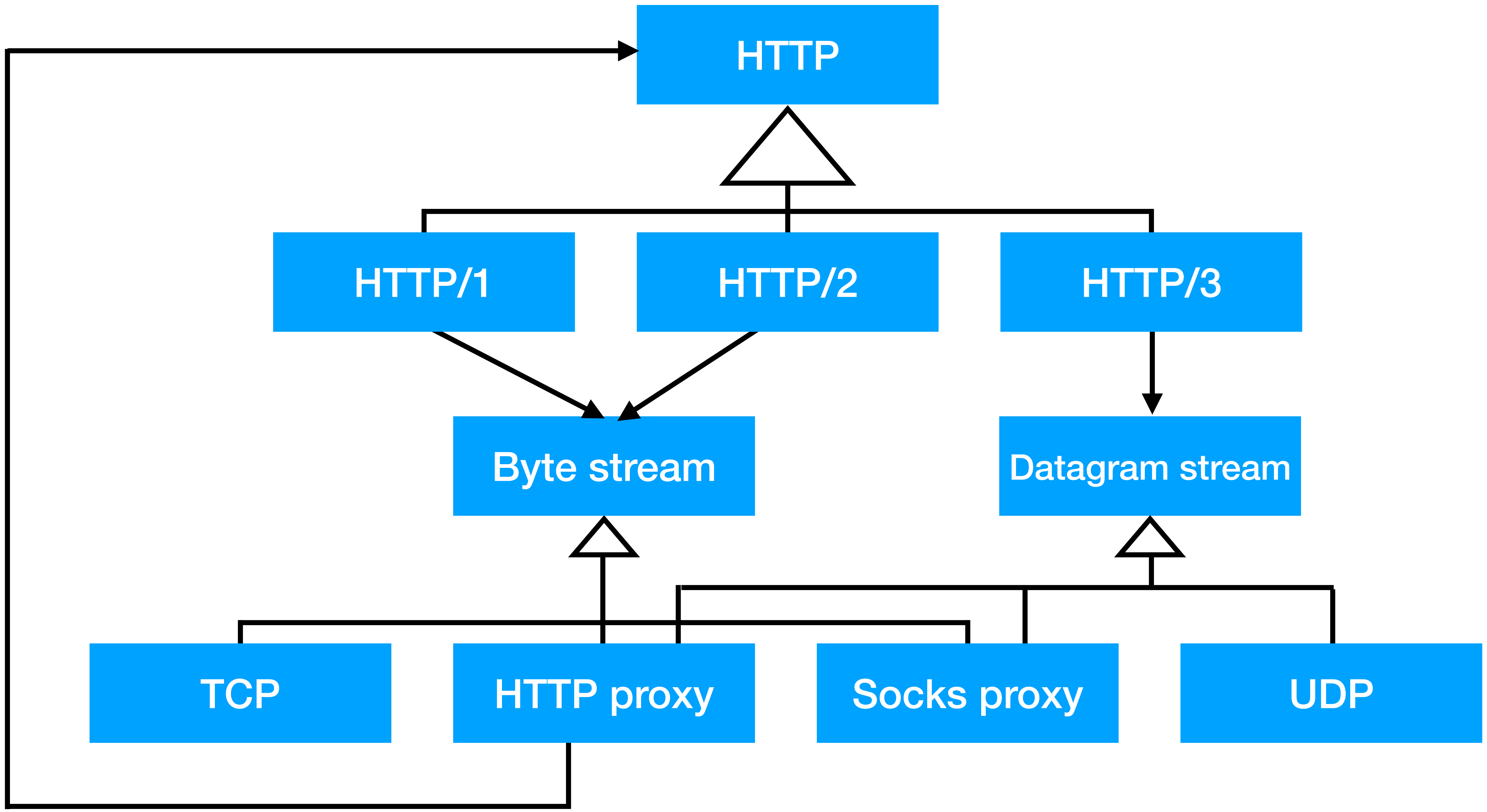
https://github.com/encode/httpx/issues/275

Summary

```
CONNECT example.com:80 HTTP/1.1

HTTP/1.1 200 Connection established

...
```

HTTP tunneling

Forget forwarding!



Protocol combinations

Proxy × origin

RFC 9110: HTTP Semantics
RFC 9112: HTTP/1.1
RFC 9113: HTTP/2
RFC 9114: HTTP/3

Interfaces everywhere

SansIO is great. Native libraries too.

How I wrote a Python client for HTTP/3 proxies

Miloslav Pojman
@MiloslavPojman

EuroPython
Dublin, 14th July, 2022

https://pojman.cz/2022/masque/