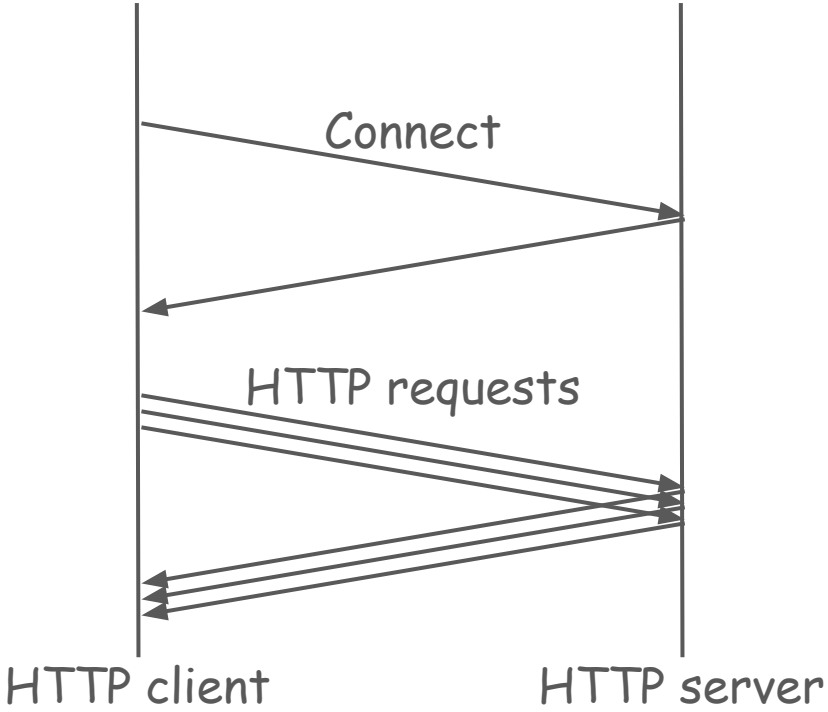
Reverse Tunnel over HTTP

draft-kazuho-httpbis-reverse-tunnel

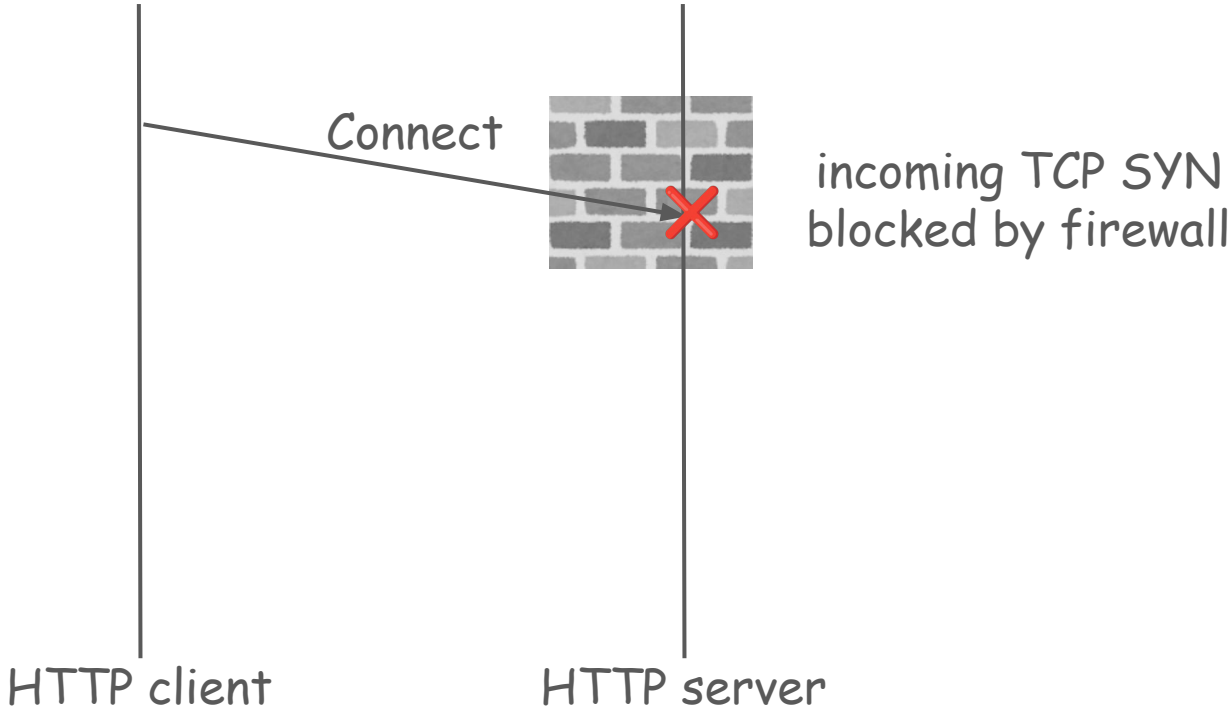
Kazuho Oku

the concept

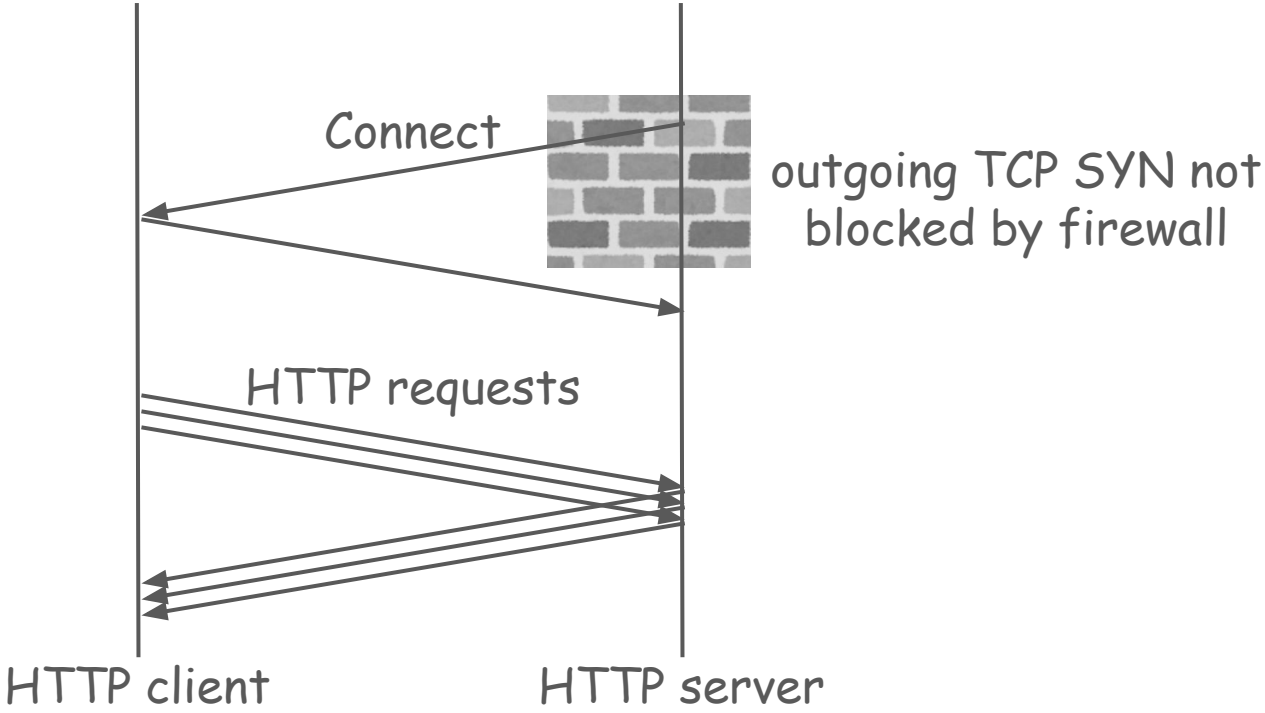
ordinary HTTP



ordinary HTTP (server behind firewall)



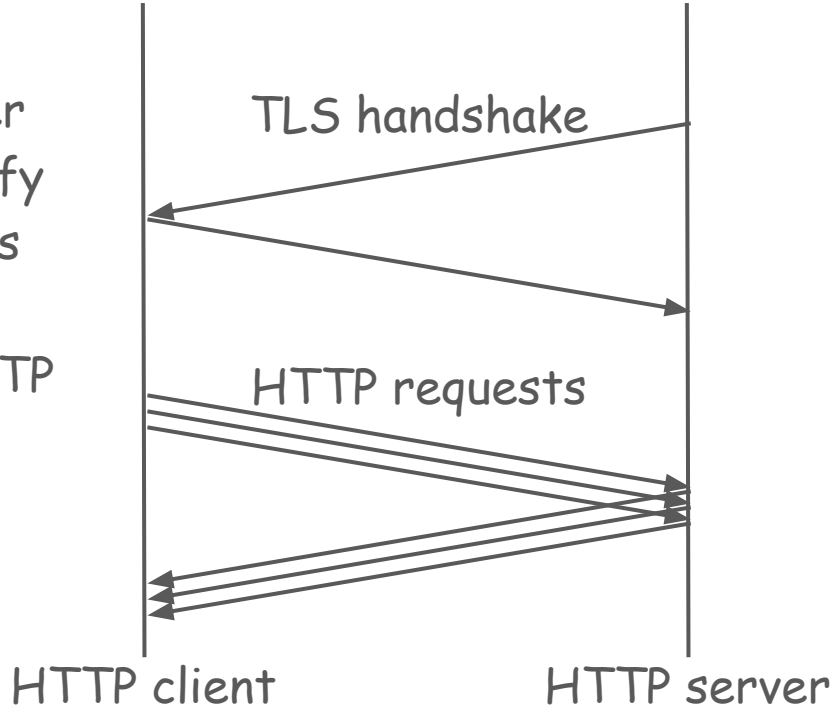
reverse HTTP



but exactly how?

"Reverse HTTP" proposal @ IETF 118

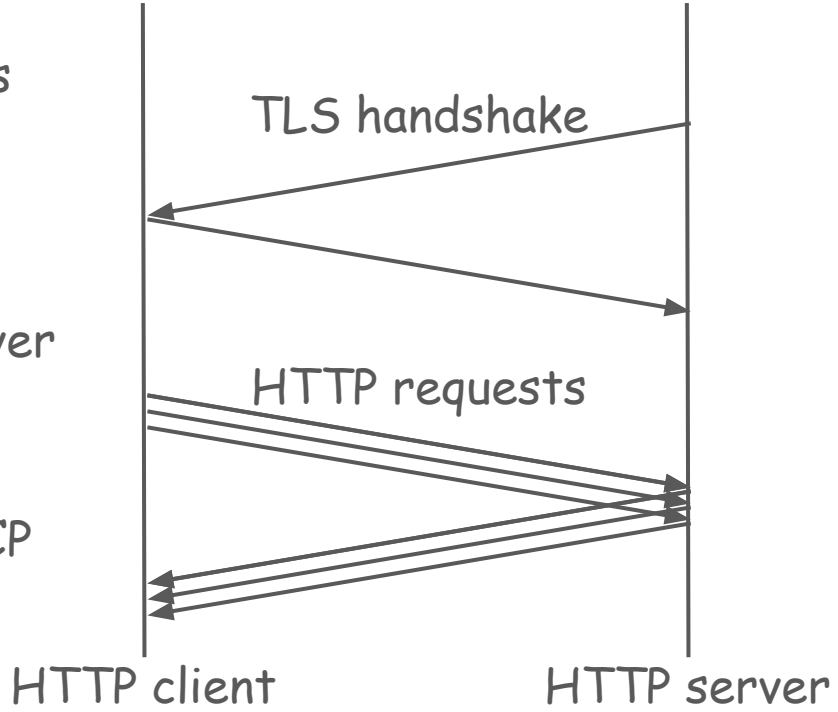
- TLS handshake carries:
  - special ALPN
  - client cert authenticates the server
  - the cert and ORIGIN frame identify HTTP resources the HTTP server is responsible for
- once handshake is done, unmodified HTTP is used (with the TCP server being the HTTP client)



"Reverse HTTP" proposal @ IETF 118

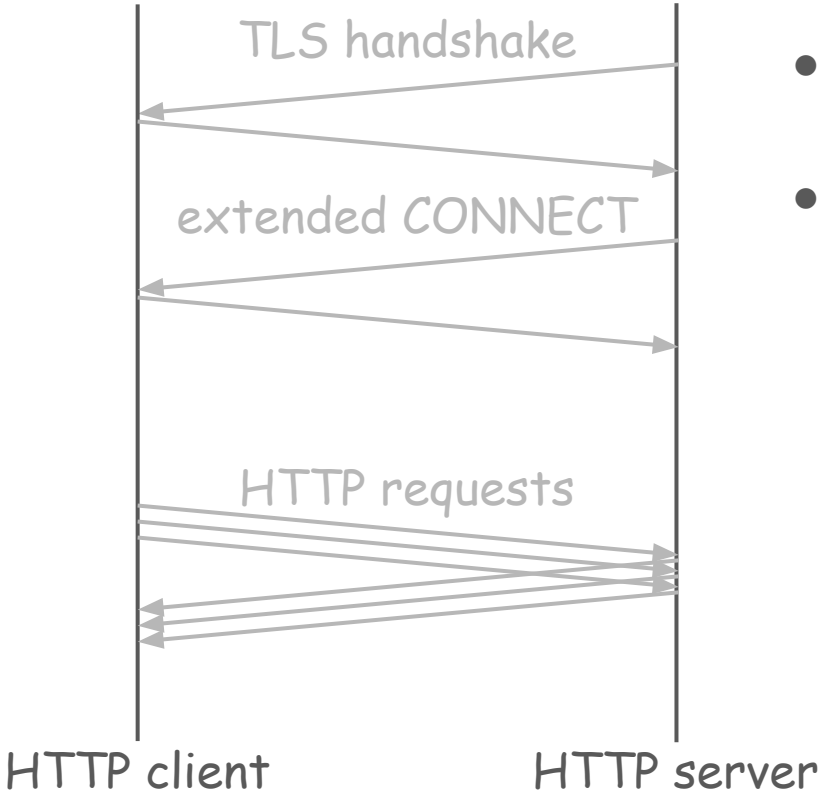
Comments at IETF 118:

- distaste to exchange tunnel parameters using TLS handshake (inflexibility)
  - mandates use of certs for authentication
  - HTTP resources for which the server is responsible are identified using certs and ORIGIN frames
- desire to use the tunnel for relaying TCP



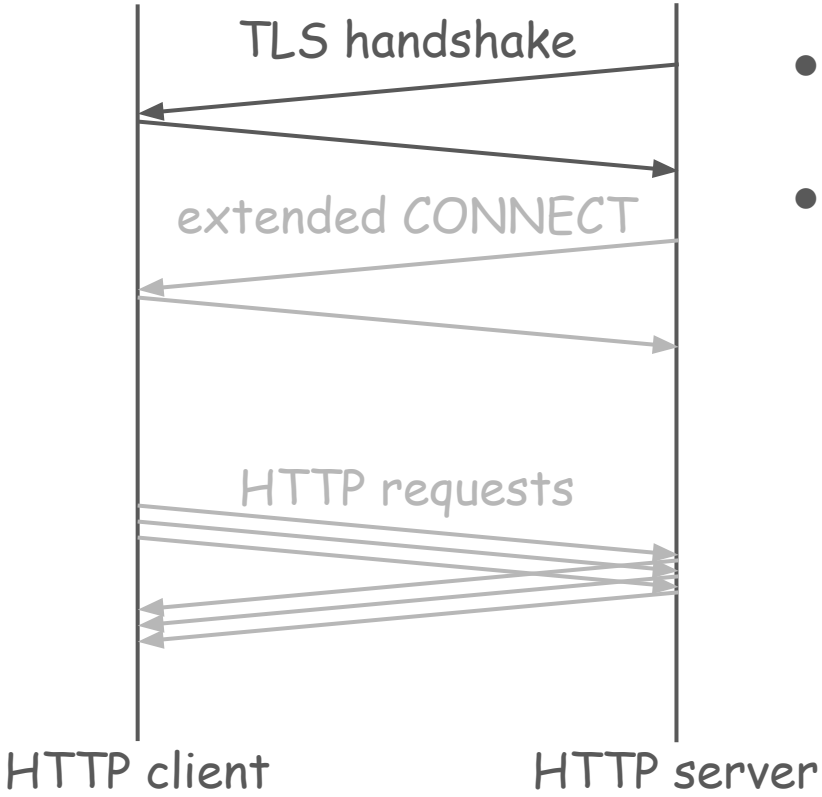
the new “Reverse Tunnel” proposal

Use HTTP to establish reverse tunnel



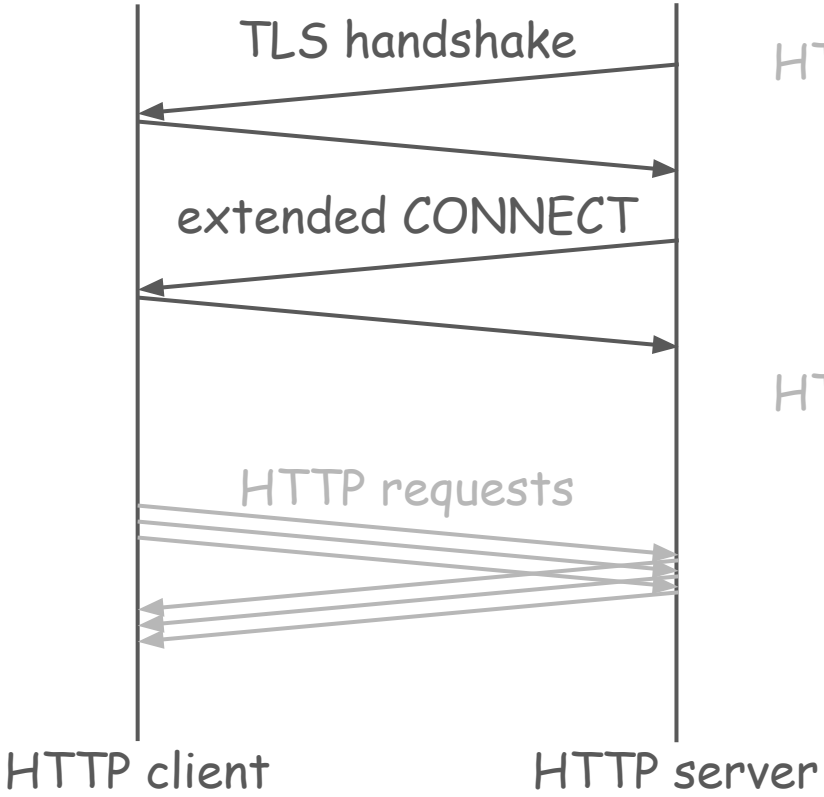
- extended CONNECT is used to establish the tunnel
- once the tunnel is established, the exchange happen on the tunnel with the roles reversed

Use HTTP to establish reverse tunnel



- TLS handshake carries ordinary ALPN: http/1.1, h2, h3
- “HTTP servers” can be authenticated using other ways than TLS client auth
  - example: basic auth

Use HTTP to establish reverse tunnel



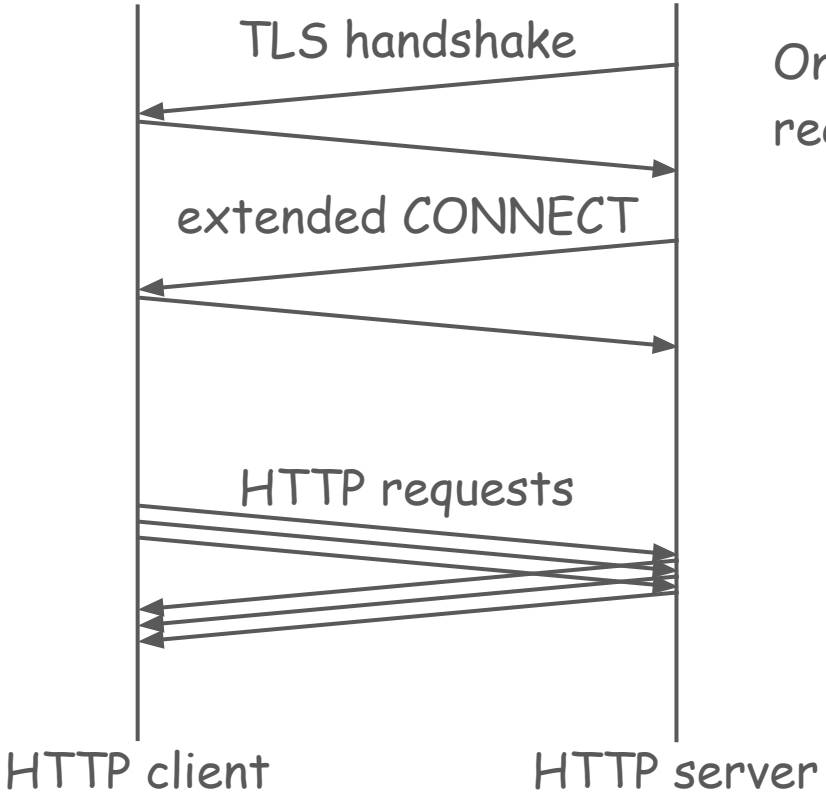
HTTP server:

GET /reverse-tunnel/of/x HTTP/1.1
Upgrade: reverse
Authorization: Basic …

HTTP client:

HTTP/1.1 101 Switching Protocols
Upgrade: reverse

Use HTTP to establish reverse tunnel

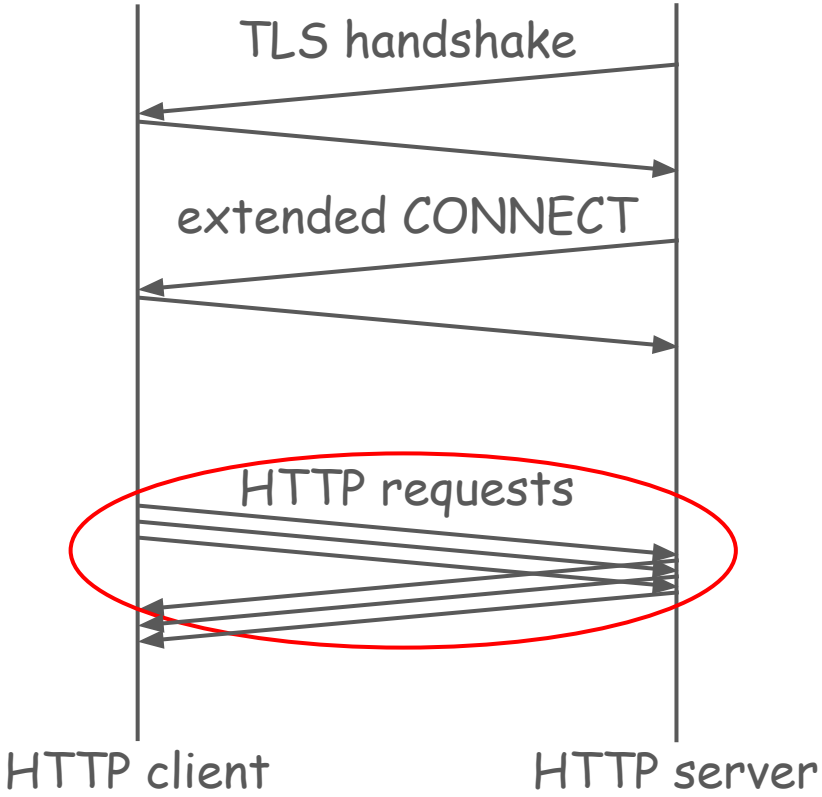


Once the reverse tunnel is established, HTTP requests start to flow from client to server

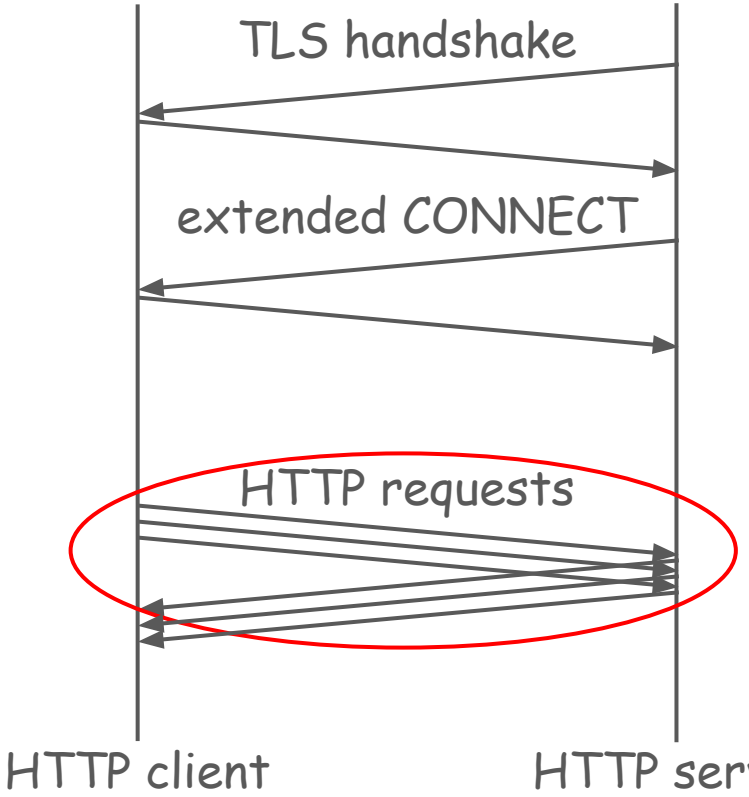
why use extended CONNECT?

- flexibility:
  - use URI (https://example.com/reverse-tunnel/of/X) to identify the resources for which the servers are responsible
    - e.g., this reverse server is responsible for path "/search?"
  - use any authentication scheme compatible with HTTP
- easier integration:
  - CDNs already provide HTTP-based APIs to the content providers
    - extended CONNECT is also HTTP
- build on top of HTTP semantics
  - rather than building one's own scheme using TLS

Which version of HTTP is it being tunnelled?



Which version of HTTP is it being tunnelled?



Option a) use TLS on top of tunnel

- cons: double encryption

Option b) use HTTP headers to negotiate

- extended CONNECT request includes:
  ALPN: h2, http/1.1
- extended CONNECT response includes:
  Selected-ALPN: h2

easy to implement, performance is guaranteed

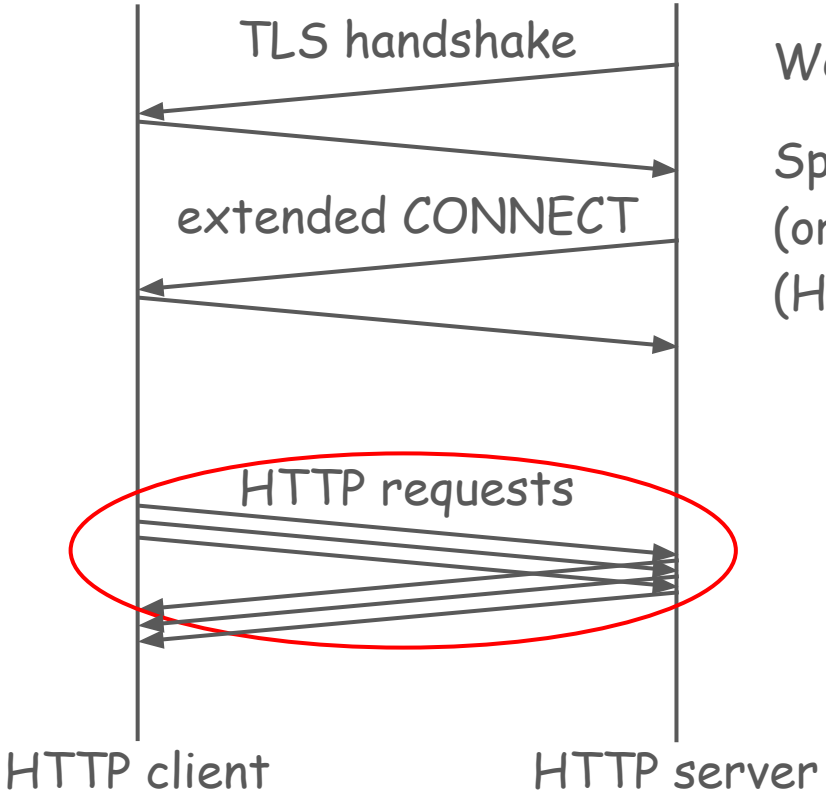
in the HTTP proxy, we want to:

- accept reverse CONNECT requests using HTTP/1.1, and
- as we send 101 Switching Protocols, move the connection state to the proxy's backend connection pool

why?

- backend connection pool can contain connections created in the normal direction and in the reverse direction, there's no need to disambiguate
- we reuse the already optimized path of HTTP proxying, once the reverse tunnel is established

What about HTTP/3?

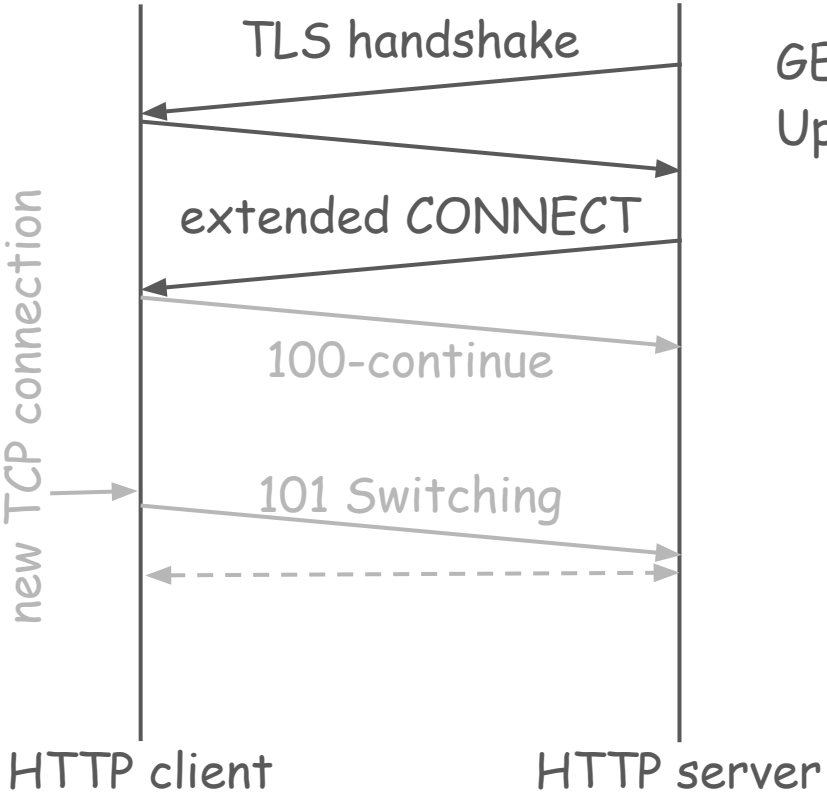


We can add support.

Specifically, we can allow use of datagrams (or capsules) on the tunnel to exchange QUIC (HTTP/3) packets.

use as a TCP relay

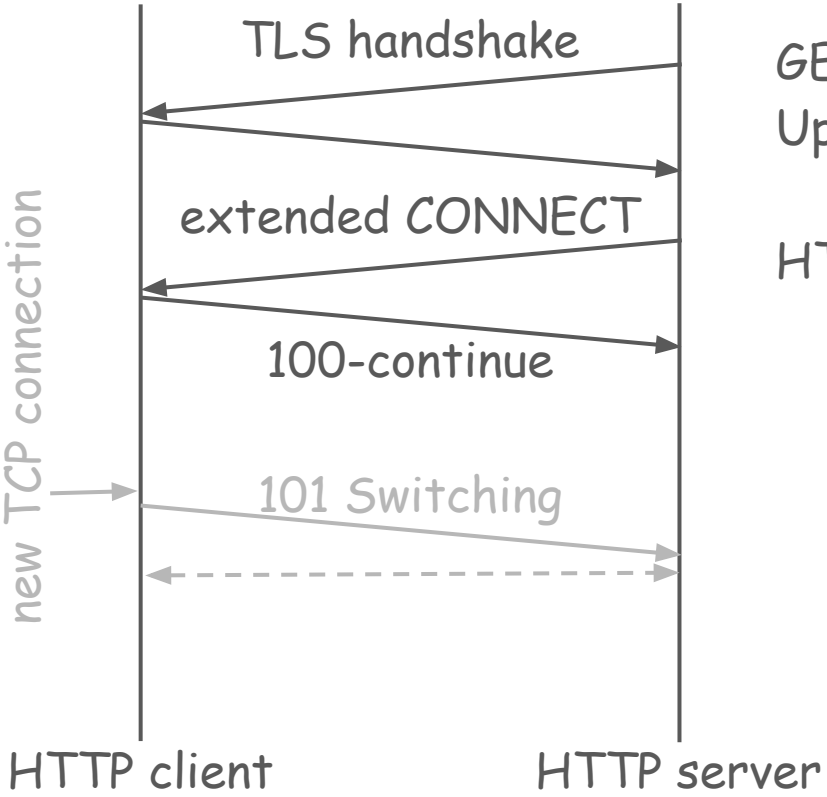
Use as a TCP relay



GET /.well-known/listen-tcp/0.0.0.0/25/ …
Upgrade: reverse

listen address

Use as a TCP relay



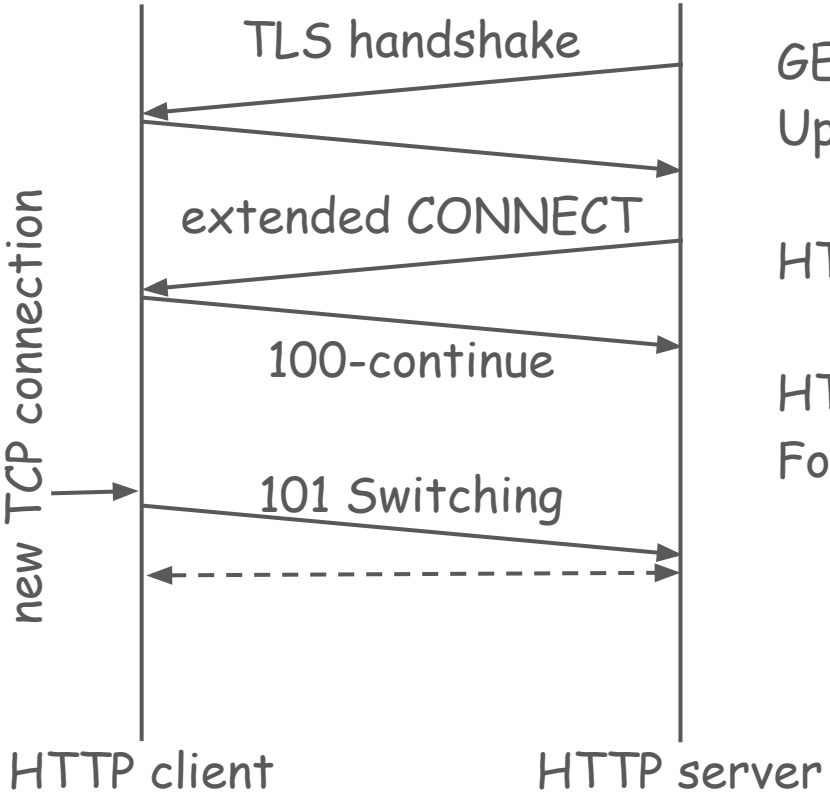
GET /.well-known/listen-tcp/0.0.0.0/25/ …
Upgrade: reverse

listen address

HTTP/1.1 100 Continue

waiting for incoming connection

Use as a TCP relay



GET /.well-known/listen-tcp/0.0.0.0/25/ …
Upgrade: reverse

HTTP/1.1 100 Continue

HTTP/1.1 101 Switching Protocols
Forwarded: for=192.0.2.43

relay connection established

Use as a TCP relay

- current semantics is accept(2), i.e.:
  - each extended CONNECT request creates a tunnel for one connection being relayed
- alternative is bind(2):
  - creation of tunnel indicates the intent to listen
  - the tunnel MUST convey H2 or H3 for multiplexing
  - for each accepted connection, HTTP client issues a CONNECT request on the tunnel and relays the TCP bytes

Questions

Questions

- Does the design look correct?
- Do we want to (need to) support HTTP3 (on QUICv1)?
- Do we need TCP relay mode?