Template-Driven HTTP CONNECT Proxying for TCP

Ben Schwartz, HTTPBIS @ IETF 116

History

- “Modernizing HTTP Proxies” presented to HTTPBIS at IETF 115
  - Covered TCP proxies and HTTP proxies
- Feedback: Separate these topics and focus on TCP proxies first

HTTP Proxying Overview

Classic HTTP CONNECT (TCP):

```
https://proxy.example

CONNECT 192.0.2.1:443 HTTP/1.1
Host: 192.0.2.1:443
…
```

- No path -> One proxy per origin
- No "Host" -> One origin per IP:port
  - Cannot use the recommended defenses against origin identity misbinding.

MASQUE (UDP, IP):

```
https://proxy.example/path{?target_
host,target_port,target,ip_proto}

:method = CONNECT
:protocol = connect-udp
capsule-protocol = ?1
:scheme = https
:authority = proxy.example
:path = /masque?
        target_host=192.0.2.1&
        target_port=443
…
```

Proposal: Template-driven TCP Transport Proxy (i.e. MASQUE for TCP)

Proxy is identified by a template:

```
https://proxy.example/tcp
{?target_host,tcp_port}
```

In HTTP/1.1:

```
GET /tcp?
    target_host=192.0.2.1&
    tcp_port=443 HTTP/1.1
Host: proxy.example:443
Connection: Upgrade
Upgrade: connect-tcp
```

In HTTP/2 & HTTP/3:

```
:method = CONNECT
:protocol = connect-tcp
:scheme = https
:authority = proxy.example:443
:path = /tcp?
        target_host=192.0.2.1&
        tcp_port=443
...
```

Closing remarks

- Useful
  - Fixes issues with Classic HTTP CONNECT on shared infrastructure.
  - More flexible support for TCP failover and Happy Eyeballs when not using implicit DNS.
  - Clarifies expectations for TCP RST and `Expect: 100-continue`.
- Convenient
  - Easy to implement and deploy alongside MASQUE.
  - No need to change client proxy configuration UIs or APIs that already take a string.
  - Can share a single template with “connect-udp” and “connect-ip”.
- Seeking adoption in HTTPBIS