Template-Driven HTTP CONNECT Proxying for TCP

Ben Schwartz, Meta Platforms Inc.
HTTPBIS @ IETF 122

Template-driven TCP Transport Proxy (i.e. MASQUE for TCP) - Now with capsules

Proxy is identified by a template:

```
https://proxy.example/tcp
{?target_host,target_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
capsule-protocol = ?1
```

In HTTP/2 & HTTP/3:

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

Changes since -01

- Removed non-capsule mode (#2943)

FINAL_SLIDE Signaling FIN vs. RST (#3000)

- Current text: Receivers assume FIN unless they get a stream, transport, or syntax error.
  - Easy receive logic, harder to send. (How do you inject a capsule syntax error?)
- PR#2949: Receivers assume RST unless they got a FINAL_DATA capsule (new).
  - Similar to WebTransport's "FIN bit".
- Kazuho and Piotr: "it might make more sense to define a frame that conveys closure and how it is closed (i.e., graceful or reset), rather than FINAL_DATA that only signals graceful shutdown".
  - "END_DATA" with a graceful/error bit
  - FINAL_DATA & "ERROR" capsules to distinguish clean/unclean and convey error info (free text?)
  - "TCP_FLAGS_DATA" capsule to carry data plus RST, FIN, PSH, and URG.
- **Need to decide.**