HTTPS and the Evil Spec Lawyer of QUIC

Them’s the rules….

RFC 3986, 3.2.3:

```
   A scheme may define a default port.  For example, the "http" scheme
   defines a default port of "80", corresponding to its reserved TCP
   port number.  The type of port designated by the port number (e.g.,
   TCP, UDP, SCTP) is defined by the URI scheme.  URI producers and
   normalizers should omit the port component and its ":" delimiter if
   port is empty or if its value would be the same as that of the
   scheme's default.
```

A long time ago, in an IETF far, far away….

RFC 2818, 2.3:

```
   When HTTP/TLS is being run over a TCP/IP connection, the default port
   is 443. This does not preclude HTTP/TLS from being run over another
   transport. TLS only presumes a reliable connection-oriented data
   stream.
```

What we currently say.... (1/2)

RFC 7230, 2.7.1:

```
   Although HTTP is independent of the transport protocol, the "http"
   scheme is specific to TCP-based services because the name delegation
   process depends on TCP for establishing authority.  An HTTP service
   based on some other underlying connection protocol would presumably
   be identified using a different URI scheme, just as the "https"
   scheme (below) is used for resources that require an end-to-end
   secured connection.  Other protocols might also be used to provide
   access to "http" identified resources -- it is only the authoritative
   interface that is specific to TCP.
```

What we currently say…. (2/2)

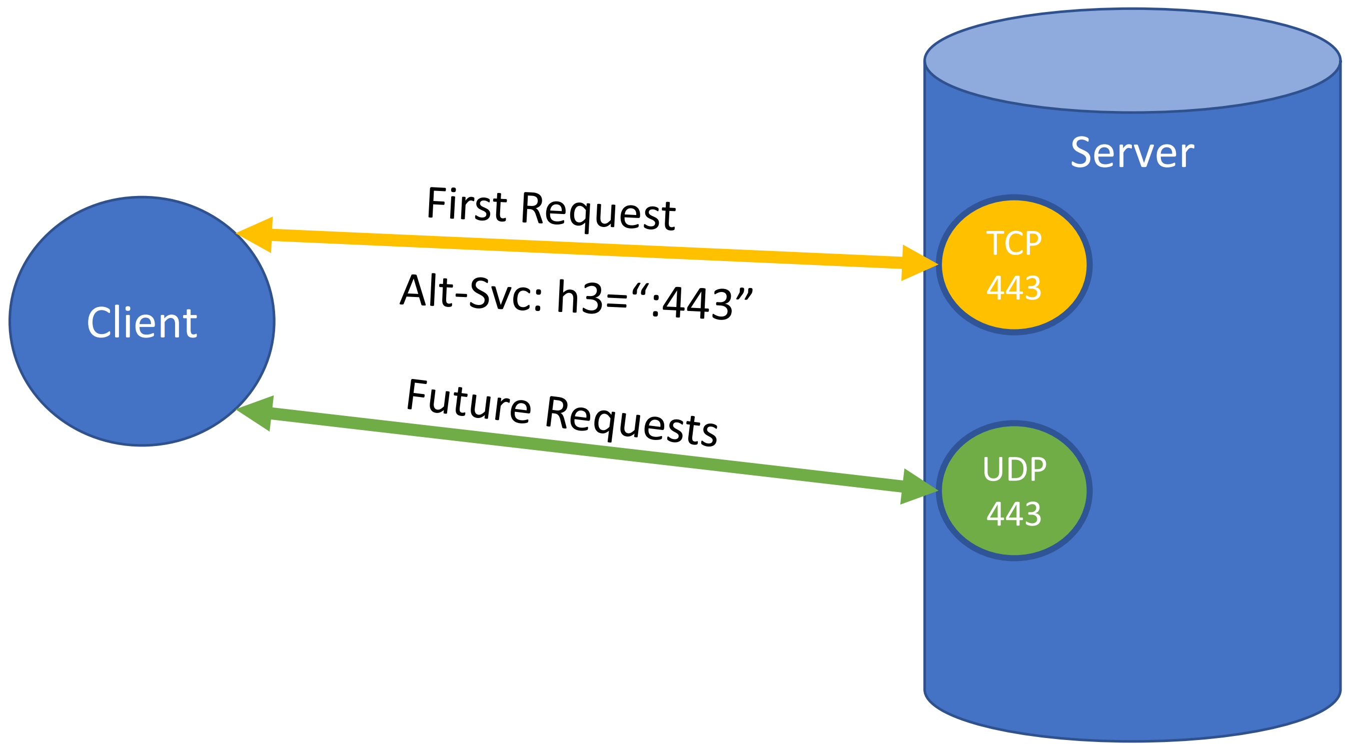
RFC 7230, 2.7.2:

```
   The "https" URI scheme is hereby defined for the purpose of minting
   identifiers according to their association with the hierarchical
   namespace governed by a potential HTTP origin server listening to a
   given TCP port for TLS-secured connections ([RFC5246]).

   All of the requirements listed above for the "http" scheme are also
   requirements for the "https" scheme, except that TCP port 443 is the
   default if the port subcomponent is empty or not given, and the user
   agent MUST ensure that its connection to the origin server is secured
   through the use of strong encryption, end-to-end, prior to sending
   the first HTTP request.
```

?

Option 1: Long Live TCP!



- RFC 7230: "The authoritative endpoint is specific to TCP"
- TCP is the bootstrap
  - Every HTTP(S) server will always have a TCP endpoint for first contact
  - Every HTTP(S) client will need a TCP stack for discovery
- Alt-Svc enables delegation to an alternative endpoint
  - Also provides vehicle for QUIC version hint
- Might define other ways to get the Alt-Svc record (e.g. DNS) to avoid the TCP connection

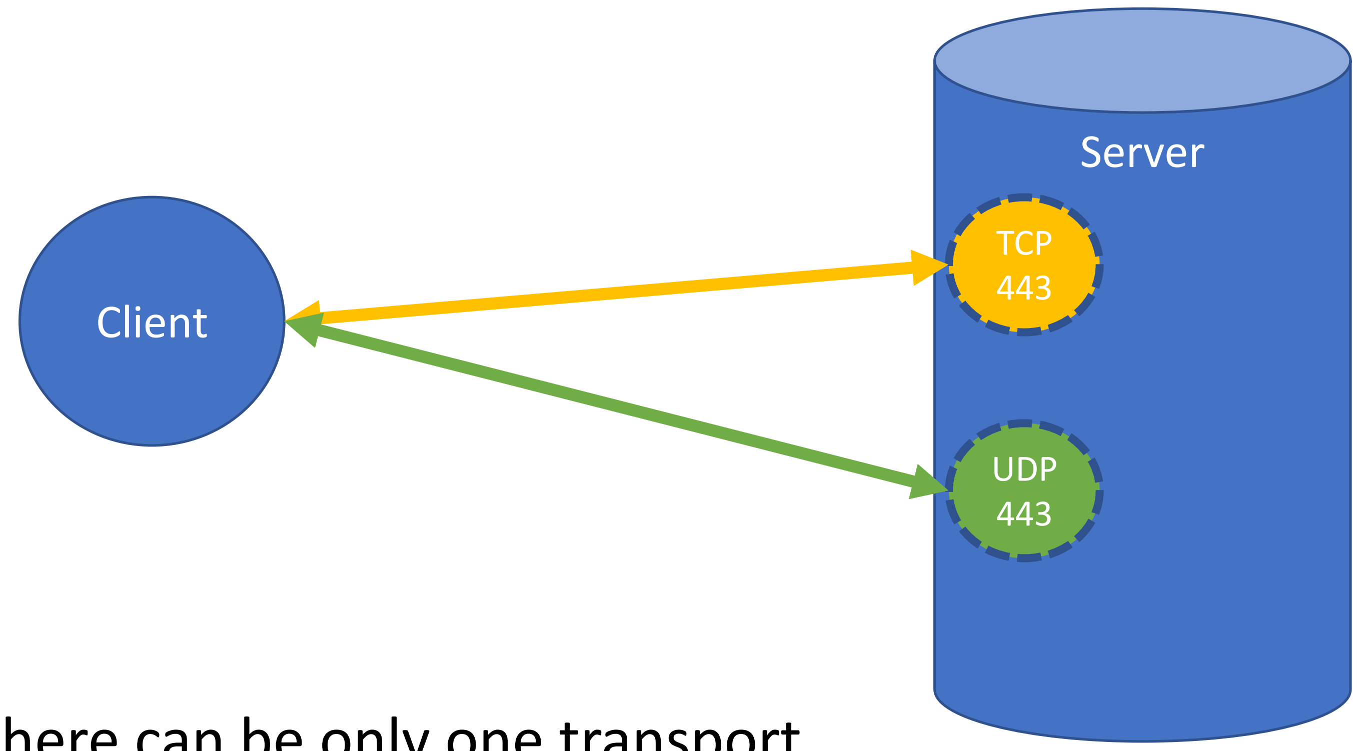
YOU ARE HERE

Option 2: ~~Mutilate~~ Decorate the URI!

```
https://www.example.com:q443/
httpq://www.example.com/
https://quic@www.example.com/
```

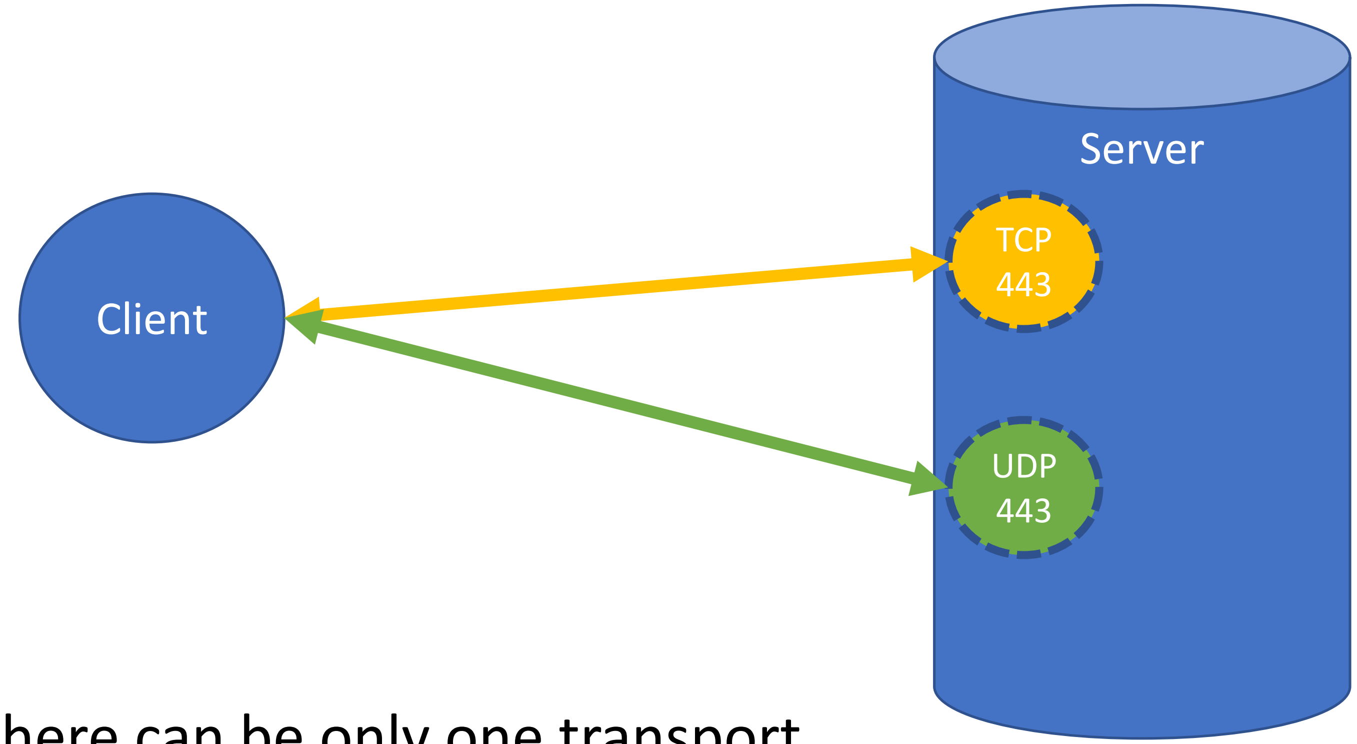
- RFC 7230: “[another] connection protocol would … be identified using a different URI scheme”
- Supports authoritative HTTP/3 endpoints
  - …and QUIC-only endpoints or clients
- Bizarre in various ways:
  - (1) violates RFC 3986, breaks URI parsers
  - (2) is a “disaster of nuclear proportions”
  - (3) redefines deprecated functionality

Option 3: Dual-Stack – it worked for IPv6!



- RFC 3986 doesn't actually *say* there can be only one transport protocol
- We're in the process of revising RFC 723X
  - …so we can change the definitions!
- New rule: "https" means TCP *or* UDP port 443
  - …and maybe "http" means TCP *or* UDP port 80?
- No QUIC version hint

Option 3: Dual-Stack – it worked for IPv6!



- RFC 3986 doesn't actually *say* there can be only one transport protocol
- We're in the process of revising RFC 723X
  - …so we can change the definitions!
- New rule: "https" means TCP *or* UDP port 443
  - …and maybe "http" means TCP *or* UDP port 80?
- No QUIC version hint
- Even Happier Eyeballs!

Will QUIC-only servers exist?

- How do we address them?

Will QUIC-only clients exist?

- How do they handle an “http(s)” URL?

Managing TCP vs. QUIC load

Does this impact security for existing servers?

Other Considerations