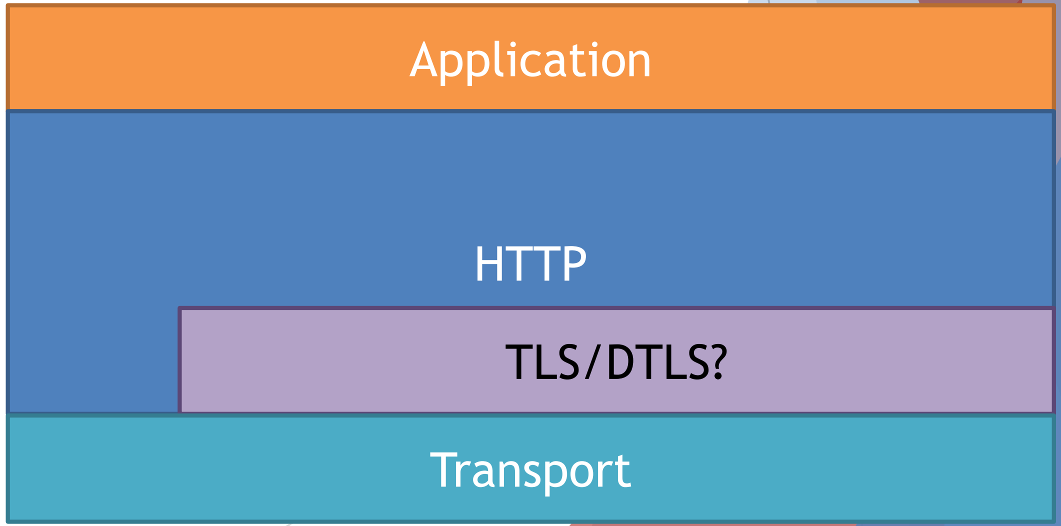
Disentangling ~~Decomposing~~ HTTP

There are lots of HTTPs

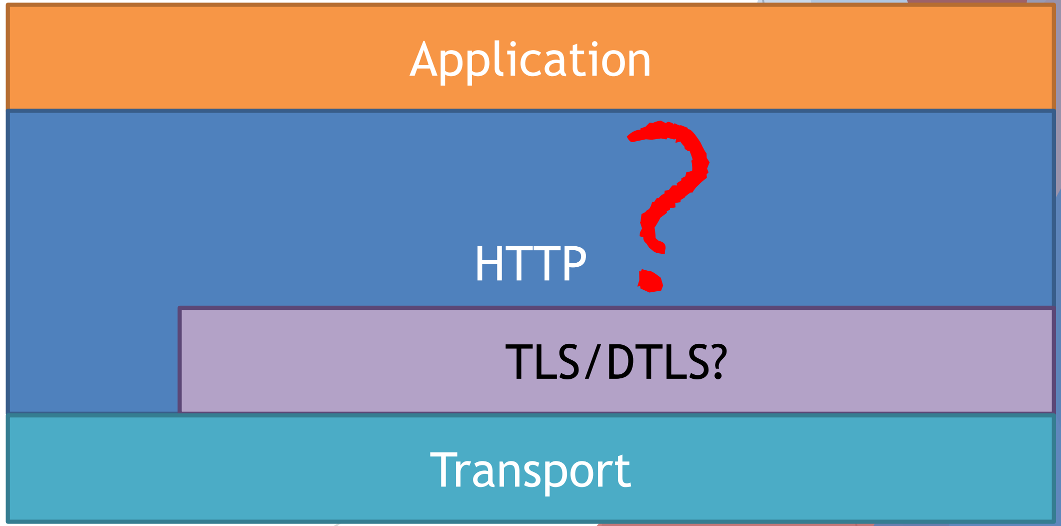
- HTTP/0.9, HTTP/1.0, HTTP/1.1
  - ASCII-ish octets over TCP
- HTTP/2
  - Binary framing layer over TCP
- HTTP/1.1 over SCTP
- HTTPU and HTTPUM
  - Subset over UDP
- CoAP
  - Super/Subset over UDP/TCP
- QUIC
  - Binary framing layer over UDP



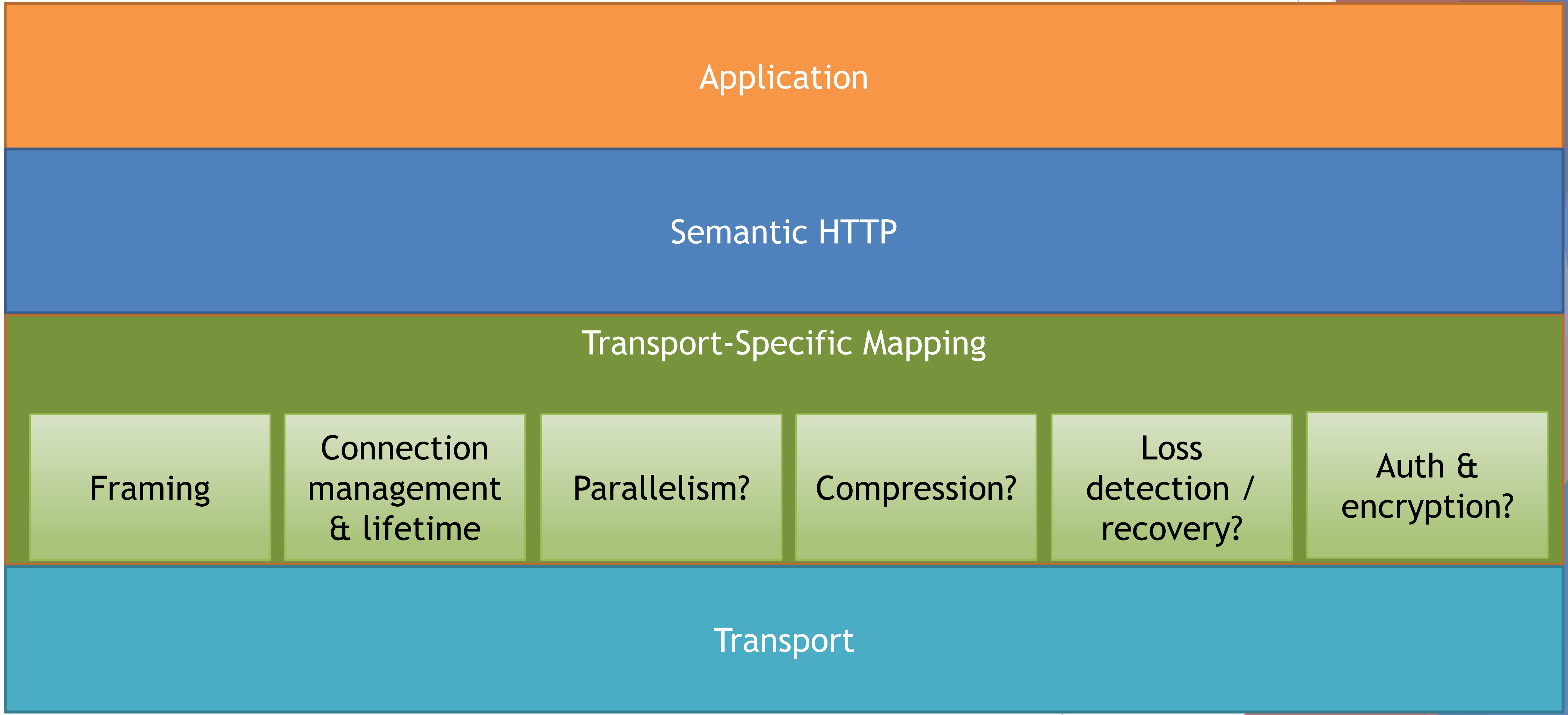
HTTP isn’t that monolithic

- Similar concepts exposed to app regardless of version
- Wildly different capabilities from transport

Does that mean the richness of TCP isn’t needed (since UDP works)?



Key Idea: Implicit middle layer


Application
Semantic HTTP
Transport-Specific Mapping
Framing
Connection management & lifetime
Parallelism?
Compression?
Loss detection / recovery?
Auth & encryption?
Transport

Middle layer: Thick or thin?

- Semantic HTTP requires certain properties
  - No transport has all of them; some have most, others very few
  - Largely unchanged HTTP/1.0 vs. HTTP/1.1 vs. HTTP/2
- Mapping HTTP to a transport requires plugging the gaps
  - Mapping defines a middle layer that implements anything the transport doesn’t provide
  - Transport + Mapping is effectively an “idealized” transport for HTTP
  - Alternative: Subset HTTP functionality to avoid the gaps
- HTTP/1.x: Simple mapping to TCP
  - ASCII-like message framing
  - Independent TCP flows to provide parallelism
- HTTP/2: Rich mapping to TCP
  - Full multiplexing layer with binary framing and multiplexing

Perils of Forgetting

- Connection: and Proxy-Connection: headers in HTTP/1.1
  - See RFC 7230 A.1.2
- CoAP’s continuing evolution
  - RFC 7252: Basic reliability over UDP/DTLS, no large messages
  - But then:
    - draft-ietf-core-block – messages bigger than a single datagram
    - draft-bormann-core-cocoa - …and congestion control
    - draft-ietf-core-tcp-tls – just use TCP!
- HTTP/2 framing layer
  - Semi-goal during design to keep the framing layer reusable by non-HTTP protocols
  - HTTP-specific concepts crept in anyway
    - Non-HTTP users would have to define a new, strikingly similar framing layer

And then there's QUIC….

```
QUIC (Quick UDP Internet Connection) is a new multiplexed and secure
transport atop UDP, designed from the ground up and optimized for HTTP/2
semantics. While built with HTTP/2 as the primary application protocol, QUIC
builds on decades of transport and security experience, and implements
mechanisms that make it attractive as a modern general-purpose transport.
QUIC provides multiplexing and flow control equivalent to HTTP/2, security
equivalent to TLS, and connection semantics, reliability, and congestion
control equivalent to TCP.
```

- Is QUIC another HTTP-over-UDP mapping?
  - Peer of HTTP/1.1, HTTP/2, HTTPU, CoAP, etc.?
- Or is QUIC another transport protocol over which HTTP *can be mapped*?
  - Peer of TCP, SCTP, UDP, etc.?
- Reality: It's currently both, in the same document.

What does it mean?

- Somewhat philosophical – no immediate actions here
  - The definition of "Semantic HTTP" is still really thin; does it matter?
  - Transports once asked for a list of services we ideally want from the transport below us. Is this the list?
- Ideas to keep in mind with our next newly-defined HTTP mapping:
  - Does QUIC belong in HTTP WG, or somewhere in Transports area?
  - Need to limit cross-contamination of HTTP concepts with mapping-internal concepts