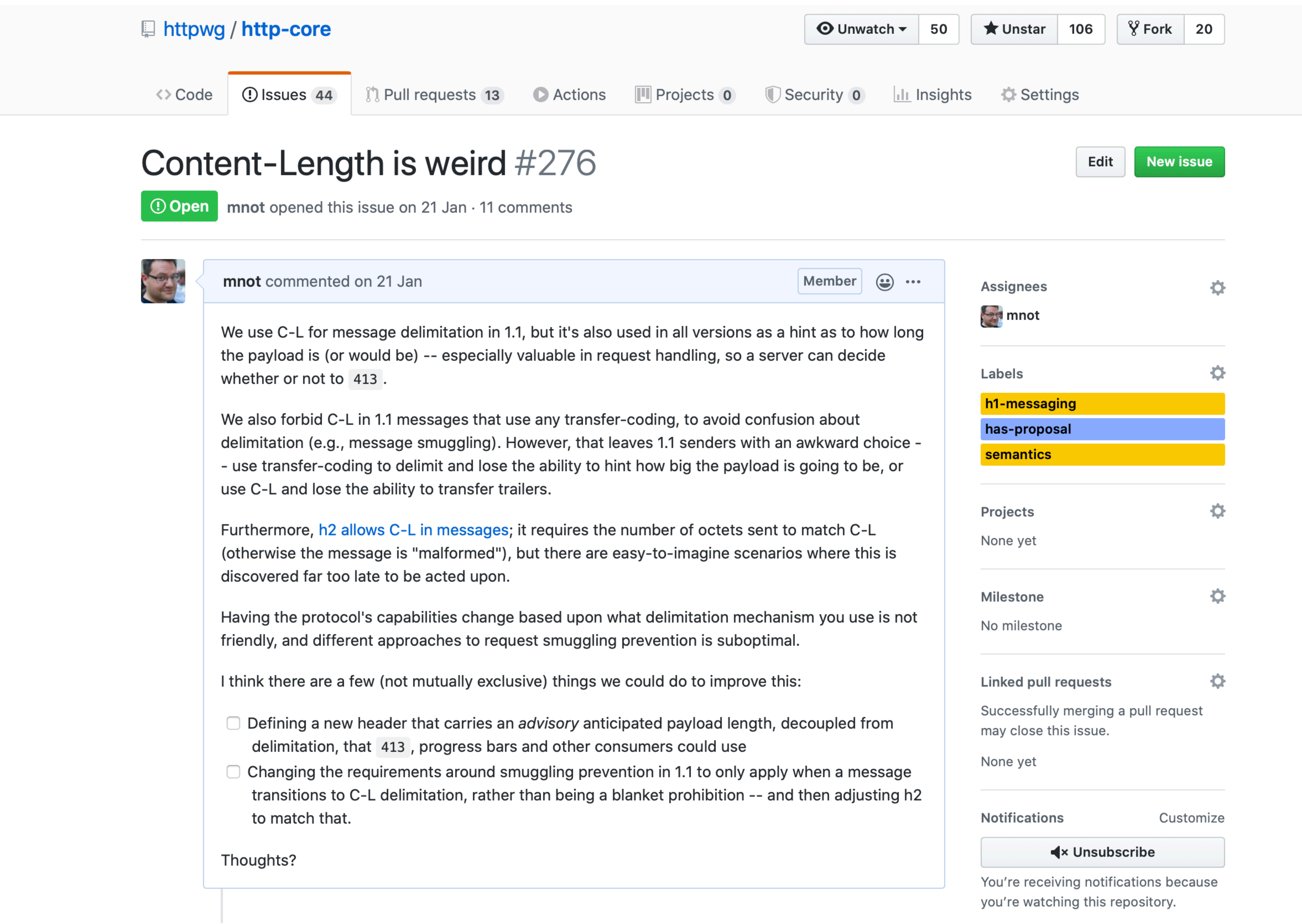
Content-Length is weird

draft-nottingham-bikeshed-length

Mark Nottingham, 26 May 2020



Content-Length is weird #276

Open mnot opened this issue on 21 Jan · 11 comments

mnot commented on 21 Jan

We use C-L for message delimitation in 1.1, but it's also used in all versions as a hint as to how long the payload is (or would be) -- especially valuable in request handling, so a server can decide whether or not to `413`.

We also forbid C-L in 1.1 messages that use any transfer-coding, to avoid confusion about delimitation (e.g., message smuggling). However, that leaves 1.1 senders with an awkward choice -- use transfer-coding to delimit and lose the ability to hint how big the payload is going to be, or use C-L and lose the ability to transfer trailers.

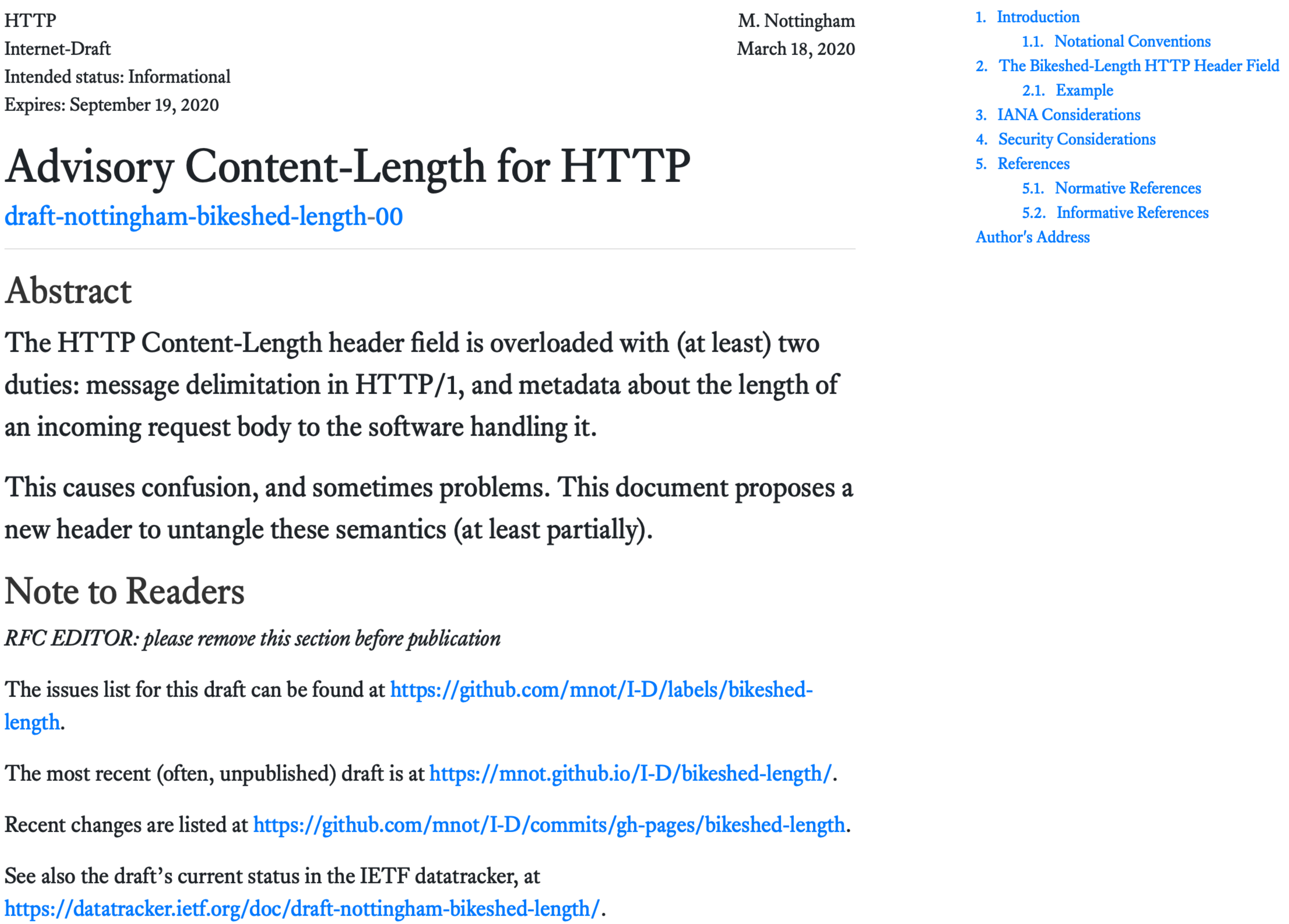
Furthermore, h2 allows C-L in messages; it requires the number of octets sent to match C-L (otherwise the message is "malformed"), but there are easy-to-imagine scenarios where this is discovered far too late to be acted upon.

Having the protocol's capabilities change based upon what delimitation mechanism you use is not friendly, and different approaches to request smuggling prevention is suboptimal.

I think there are a few (not mutually exclusive) things we could do to improve this:

- □ Defining a new header that carries an *advisory* anticipated payload length, decoupled from delimitation, that `413`, progress bars and other consumers could use
- □ Changing the requirements around smuggling prevention in 1.1 to only apply when a message transitions to C-L delimitation, rather than being a blanket prohibition -- and then adjusting h2 to match that.

Thoughts?



HTTP
Internet-Draft
Intended status: Informational
Expires: September 19, 2020

M. Nottingham
March 18, 2020

Advisory Content-Length for HTTP

draft-nottingham-bikeshed-length-00

Abstract

The HTTP Content-Length header field is overloaded with (at least) two duties: message delimitation in HTTP/1, and metadata about the length of an incoming request body to the software handling it.

This causes confusion, and sometimes problems. This document proposes a new header to untangle these semantics (at least partially).

Note to Readers

RFC EDITOR: please remove this section before publication

The issues list for this draft can be found at https://github.com/mnot/I-D/labels/bikeshed-length.

The most recent (often, unpublished) draft is at https://mnot.github.io/I-D/bikeshed-length/.

Recent changes are listed at https://github.com/mnot/I-D/commits/gh-pages/bikeshed-length.

See also the draft's current status in the IETF datatracker, at https://datatracker.ietf.org/doc/draft-nottingham-bikeshed-length/.

Content-Length is weird

because it serves more than one purpose

- **HTTP/1.x message delimitation**
  - Extremely security sensitive, so
  - Typically NOT under direct application control
  - Only used in 1.x
- **Setting peer expectations about size**
  - e.g., deciding whether to accept a POST body
  - e.g., showing download progress
  - Not version-specific
  - Great precision not needed

Content-Length

needs careful guardrails

- HTTP/1 forbids C-L in any message with Transfer-Encoding
- Even when the next hop isn't HTTP/1, you need to consider that one beyond it might be.
- H2 and H3 require C-L in message to match bytes on wire
    - ... but recipients may be too late to enforce this

Proposal:

Separate these uses

- New header field for conveying **advisory length**
  - Name TBD
  - Same syntax as Content-Length
    - ... but specified as a SF-Integer
  - No constraints about when it can, can't be sent, etc.
  - Presumption is that recipients would use it to inform decisions, while keeping an eye on the actual number of bytes seen
  - Would help chunked transfer-encoding of requests

Questions

for the WG

- Is standardising this header field helpful?
- Should it be in the HTTP Semantics document, or separate?