Compression in Waka

Roy T. Fielding, Ph.D
Senior Principal Scientist, Adobe

Waka

A replacement for HTTP (mostly designed in 2001)

- Token-based, length-delimited syntax*
  (at base level without compression)
- Self-descriptive messages

Interleaved message (meta)data packets:

- Up to 64 channels per connection
- Up to 63 payload streams per message

Complete transport independence

- TCP, UDP, SCTP, TLS, QUIC, multicast, ...

Restarting development now

Waka Base Syntax

Uniform syntax

- Regardless of message type, explicit direction
- Padding allowed for 32/64bit alignment

Self-descriptive

- Explicit typing for message structure, fields
- Indication of mandate and scope of fields
- Association of metadata (control, resource, rep.)
- Premature termination of request or response

Efficient and Extensible

- Tokens for all standard elements
- A URI reference can be used in place of any token
- Interleaved data and metadata delivery
- ~~Macros (client-defined syntax short-hand)~~*

Wakli Compression

Waka is already a tight syntax, but we can do better

- client-directed macro storage is too easy for DoS
- HPACK-style compression is complex, HOL-blocking
- dynamic compression reveals information under TLS
- no one-size-fits-all solution for mobile/IoT space

Design idea for Wakli: similar to Brotli, but

- with a static dictionary organized into levels
- 0, 8192, 32768, 131072, 524288, 2MB, 8MB, 32MB
- message signals level used & max level accepted
- initial message starts at level 0 (or last used)
- sender chooses tradeoff of compression/memory
- recipient can reject unsupported or excessive use
- data can be selectively excluded from compression

What I Need

① Sanity Check

② Real HTTP message traces for corpus
(not JSON, de-identified with %x0C)

③ Review future specification drafts at
https://github.com/dwid-org/waka

④ Develop and test implementations

⑤ Analyze HTTP ⇒ waka ⇒ wakli ratios