moqpack

IETF 125

Alan Frindell, Meta

MOQT basics in 60s

- SETUP exchange on pair of unidirectional streams
- Requests -- eg SUBSCRIBE/PUBLISH -- sent by either endpoint on bidi streams
- Requests contain key/value parameters; some could be long / repeated
  - Examples: Track Names, Auth Tokens
- Sounds familiar right?

Important Differences from HTTP/3

- Requests have a Request ID that is ~ 1:1 to the stream
  - WebTransport does not expose Stream IDs
- Track Names consist of two parts: Namespace and Name
  - Namespace ~ directory name -- tuple of (length, bytes) pairs
  - Name ~ file name -- (length, bytes)
- Parameters have only integer keys
- Parameter values are bytes - key determines semantics

Let's Compress!

- Goal to re-use as much of QPACK as possible
- Correct some H3 whiffs
  - Make it entirely optional
    - QPACK is a barrier to a simple H3 implementation
  - Yeet Huffman -- compression benefits doesn't justify the CPU cost

What's the Same?

- SETUP / SETTINGS
  - TABLE_CAPACITY, BLOCKED_STREAMS
- Unidirectional Encoder/Decoder Streams
- Messages carry compressed blocks in QPACK format (Insert Count, Base, etc)
- Reused existing instructions across blocks, encoder, decoder
- "Never Index" semantics

What's different?

Reinterpretation of "static table references"

- MOQT has no values common enough to warrant a true table
- A static table name reference index *directly* encodes the MOQT parameter key
- Several instructions are prohibited:
  - Encoder Stream:
    - **Insert With Dynamic Name Reference** - no dynamic names
    - **Insert With Literal Name** - names represented as static table indexes
  - Compressed Blocks:
    - **Indexed Field Line with Static Table** - no static values
    - **Literal Field Line With Dynamic/Post-Base Name Reference** - no dynamic names
    - **Literal Field Line With Literal Name** - names represented as static table indexes

Allowed Instructions

- **Encoder Stream**
  - Insert With Static Name Reference
  - Set Dynamic Table Capacity
  - Duplicate
- **Compressed Blocks**
  - Literal Field Line With Static Name Reference
  - Indexed Field Line with Dynamic Table
  - Indexed Field Line with Post-Base Index
- **Decoder Stream**
  - Insert Count Increment
  - Section Acknowledgment
  - Stream Cancellation

What's different?

- Table Size Computation - all names are 4 bytes
- Decoder "Stream IDs" replaced with "Request IDs" -- required for WebTransport
- Pseudo params and Track Name reassembly
  - TRACK_NAMESPACE_ELEMENT, TRACK_NAME (bytes)
  - TRACK_NAMESPACE_SET (length, bytes)+
  - ELEMENT and SET can be used to compress "partial paths" in a way QPACK cannot
    - (long, path, to, resource1) and (long, path, to, resource2) can be compressed by inserting (long, path, to) as a TRACK_NAMESPACE_SET

Parameters with integer values

- MOQT recently changed parameters to be TV -- the receiver MUST understand the keys in order to parse
- For MOQPACK, left instructions as T**L**V
  - Key=DELIVERY_TIMEOUT, Value=500
    Literal with Static Name Reference: Key (0x8), Length (2), Value(moqt-varint(500))
- If we make MOQPACK TV aware, could encode the *value* as the length
  - Literal with Static Name Reference: Key (0x8), Value (qpack-varint(500)).
- TLV MOQPACK more nimble to handle extensions that define new keys *but* introduces length-mismatch errors and is slightly less efficient.

Discussion

- Design Feedback?
  - Optionality, removing Huffman OK? Partial path compression? TV/TLV?
- Is this a good use of QPACK or bad use of QPACK?
  - Similar but not the same, unlikely anyone could use it "off the shelf"
- Is making an protocol *agnostic* key/value compression protocol useful?
- General QPACK thoughts?