QPACK

draft-bishop-quic-http-and-qpack
Resilient Header Compression

**HPACK**
- Inserts always append to the dynamic table
- On-stream operations modify the table
- Table size managed implicitly
  - If the table overflows, drop oldest value
  - Indices change over time

**QPACK**
- Inserts are to an explicit index
- Table management on a dedicated stream
  - On-stream operations never modify table state
- Table size managed by explicitly deleting entries
  - If the table overflows, kill the connection
  - Indices are consistent over time

<table>
<thead>
<tr>
<th>Index</th>
<th>Horizon</th>
<th>Stream Refs</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Reset-Independent

HPACK

Insert+Use

Use

RST_STREAM

No control stream can ever be reset safely!

QPACK

Insert

Use

Use

RST_STREAM

Protect critical content on a single shared control stream
### Order-Independent

<table>
<thead>
<tr>
<th>Index</th>
<th>Horizon</th>
<th>Stream Refs</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

M: **KEY=VALUE at INDEX**

5: **Reference INDEX**

9: **Reference INDEX**

M: **Delete INDEX (From 0: 5, 9)**
## Insert Reordering

<table>
<thead>
<tr>
<th>Index</th>
<th>Horizon</th>
<th>Stream Refs</th>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

### Instructions

- **5: Reference INDEX**

- **M: KEY=VALUE at INDEX**

- **9: Reference INDEX**

- **M: Delete INDEX (From 0: 5, 9)**

### Notes

- References block if the field isn’t defined yet
Delete Reordering

- Deletes specify the streams with references
- If not all references have arrived yet, deletes deferred until the last one arrives
- Insert/Delete on same stream, so Delete can’t arrive before Insert
When can you delete?

• Delete contents:
  • For headers and trailers separately:
    • Starting value (Horizon)
    • List of references since Horizon

• Delete can complete when these streams have been processed:
  • All streams before Horizon
  • All streams listed

• Encoder considers space freed only once decoder declares delete completed
• Initially looks like you can cause the decoder to remember arbitrarily-long lists of stream IDs
• Memory consumption attack?
...or not.

- Horizon value allows lists to be condensed as much as either party chooses
- Used to say “no knowledge prior to stream...”
A land of trade-offs...

- Shared code with HTTP/2 versus best use of QUIC
- Design effort around reordering versus frequency of reordering
  - Google suggests that reordering is rare, but has noticeable impact when it occurs
- Flexibility to handle reset streams versus difficulty of recovery